EMNLP 2010

Conference on
Empirical Methods in Natural Language Processing

Proceedings of the Conference

9-11 October 2010

USB memory sticks produced by
Omnipress Inc.
2600 Anderson Street
Madison, WI 53707
USA

Order copies of this and other ACL proceedings from:

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

ISBN 978-1-932432-86-2 / 1-932432-86-8

Dedicated to

Fred Jelinek

(18 November 1932 – 14 September 2010)

Preface

Welcome to EMNLP 2010, Conference on Empirical Methods in Natural Language Processing! The conference is organized by SIGDAT, the Association for Computational Linguistics' special interest group on linguistic data and corpus-based approaches to NLP, and is held this year as a stand-alone conference at the MIT Stata Center, Massachusetts, USA on October 9-11. [1]

EMNLP 2010 received 500 submissions, a new record for the conference. The program committee was able to accept 125 papers in total (an acceptance rate of 25%). Among them, 70 of the papers (14%) were accepted for oral presentations, and 55 (11%) for poster presentations. The PC, which consists of 18 area chairs and 460 PC members from Asia, Europe, and North America, worked together to create a strong program with high quality oral and poster presentations and enlightening invited talks.

First and foremost, we would like to thank the authors who submitted their work to EMNLP 2010. The sheer number of submissions reflects how broad and active our field is. We are deeply indebted to the area chairs and the PC members for their hard work. They enabled us to make a wonderful program and to provide valuable feedback to the authors. We are very grateful to our invited speakers Kevin Knight, Andrew Ng and Amit Singhal, who kindly agreed to give talks at EMNLP. Many thanks to local arrangements chair, Regina Barzilay, who has made the conference smoothly held at the wonderful venue of MIT; the publications chair, Eric Fosler-Lussier, who put this volume together with assistance from Preethi Jyothi and Rohit Prabhavalkar; best paper award committee chair, Jason Eisner, who lead the effort of selecting the best papers. Special thanks to David Yarowsky and Ken Church of SIGDAT, as well as Jason Eisner, Philipp Koehn, Rada Mihalcea, who provided much valuable advice and assistance in the past months. David Yarowsky also worked on the important issues of invitation letters and travel grants. We are most grateful to Priscilla Rasmussen who helped us with various logistic and organizational aspects of the conference. Rich Gerber and the START team responded to our questions quickly, and helped us manage the large number of submissions smoothly; we would like to thank them as well.

To enhance the quality of the conference, the program committee made a number of new efforts and continued some existing good practices in PC management, paper selection, and conference participation.

First, a strict process for selecting PC members was set up. PC members were first nominated by the area chairs; PC co-chairs then carefully checked whether they were qualified as reviewers by looking at their publications and academic experience. Second, a best reviewer award was created for the first time in this conference. After the reviewing process, 86 outstanding PC members were selected as the "best reviewers" as noted in the proceedings: they submitted all the reviews on time, made detailed, constructive, and helpful comments, and actively participated in the paper discussion when necessary. These recipients were first nominated by the area chairs and then endorsed by the PC co-chairs. The awards are a recognition of the reviewers who really did hard work in reviews.

In paper selection, we created an author response period, as in the two previous EMNLP conferences. Authors were able to read and respond to the reviews of their papers before the program committee made a final decision. They were asked to correct factual errors in the reviews and answer questions

[1] Conference web site: http://www.lsi.upc.edu/events/emnlp2010/.

raised in the reviewer comments. In some cases, reviewers changed their scores in view of the authors' response; the area chairs read all responses carefully prior to making recommendations for acceptance. As PC co-chairs, we did our best effort in final paper selection. The area chairs made accept/reject suggestions to us on the papers in their areas. We carefully examined all the cases, discussing with the area chars the submissions on which we had divergent opinions. In some cases, the original suggestions by the area chairs were reversed after the discussions. The final paper selection was based on the consensus of the program committee. The accepted papers were further classified into oral and poster types, based on recommendations by the area chairs and discussions between the area chairs and us. Those papers that are suitable for presentations for the general audience (novel, inspiring, widely applicable, etc.) were selected as oral. After the paper notification, an independent committee for best paper awards led by Jason Eisner was created. From the candidates nominated by the PC, the committee performed independent reviews of the papers and made the best paper award selection, as listed in the proceedings.

Since EMNLP is held in the US, many participants needed visas to attend the conference. In order to help authors in the application for visas, a new procedure was implemented to anticipate the submission of invitation letters. The information on whether the authors needed a visa was collected at the paper submission time. The authors whose papers had average scores above a certain threshold were sent invitation letters by David Yarowsky, even before paper acceptance notification. In this way, the authors were given more time for their visa applications.

The success of a conference is really a result of the great efforts of everybody involved. We hope that you enjoy the conference at the fantastic building of MIT Stata Center!

Hang Li and Lluís Màrquez
EMNLP 2010 Program Co-Chairs

Program Co-chairs

Hang Li, Microsoft Research Asia
Lluís Màrquez, Technical University of Catalonia

Area Chairs

Eneko Agirre, University of the Basque Country
Kalina Bontcheva, University of Sheffield
Antal van den Bosch, Tilburg University
David Chiang, USC Information Sciences Institute
Jennifer Chu-Carroll, IBM Research
Evgeniy Gabrilovich, Yahoo! Research
Jianfeng Gao, Microsoft Research
Julio Gonzalo, UNED
Nancy Ide, Vassar College
Haizhou Li, Institute for Infocomm Research
Qun Liu, ICT, Chinese Academy of Sciences
Yusuke Miyao, National Institute of Informatics
Bo Pang, Yahoo! Research
Dragomir Radev, University of Michigan
Noah Smith, Carnegie Mellon University
Kristina Toutanova, Microsoft Research
Yoshimasa Tsuruoka, JAIST
Yi Zhang, University of California, Santa Cruz

Local Arrangements Chair

Regina Barzilay, Massachusetts Institute of Technology

Publications Chair

Eric Fosler-Lussier, The Ohio State University

Best Paper Award Committee

Jason Eisner, Johns Hopkins University (Chair)
Sharon Goldwater, University of Edinburgh Kevin Knight, University of Southern California
Dan Roth, University of Illinois, Urbana-Champaign
Ben Taskar, University of Pennsylvania
Jun'ichi Tsujii, University of Tokyo
Hanna Wallach, University of Massachusetts, Amherst

Reviewers

Hua Ai, Jan Alexandersson, Enrique Alfonseca, Enrique Amigó, Ion Androutsopoulos, Shlomo Argamon, Javier Artiles, Necip Fazil Ayan, Jing Bai, Collin Baker, Timothy Baldwin, Rafael Banchs, Carmen Banea, Shenghua Bao, Marco Baroni, Roberto Basili, Beata Beigman Klebanov, Núria Bel, Anja Belz, José-Miguel Benedí, Sabine Bergler, Mikhail Bilenko, John Blitzer, Phil Blunsom, Elizabeth Boschee, Jordan Boyd-Graber, SRK Branavan, Martin Braschler, Eric Breck, Christopher Brewster, Chris Brockett, Razvan Bunescu, Bill Byrne, Michael Cafarella, Aoife Cahill, Lynne Cahill, Chris Callison-Burch, Nicoletta Calzolari, Yunbo Cao, Claire Cardie, Andrew Carlson, Marine Carpuat, Xavier Carreras, John Carroll, Francisco Casacuberta, Asli Celikyilmaz, Joyce Chai, Raman Chandrasekar, Ming-Wei Chang, Yi Chang, Ciprian Chelba, Wenliang Chen, Colin Cherry, Jen-Tzung Chien, Hori Chiori, Yejin Choi, Ken Church, Massimiliano Ciaramita, Chris Cieri, Alex Clark, Stephen Clark, James Clarke, Paul Clough, Shay Cohen, Trevor Cohn, Nigel Collier, Michael Collins, Anna Corazza, Mark Core, Marta Costa-jussà, Josep M. Crego, Dan Cristea, Andras Csomai, Silviu-Petru Cucerzan, Hang Cui, Walter Daelemans, Robert Dale, Bob Damper, Cristian Danescu-Niculescu-Mizil, Hoa Dang, Van Dang, Sajib Dasgupta, Hal Daumé III, Adrià de Gispert, Guy De Pauw, Maarten de Rijke, Steve DeNeefe, John DeNero, Thierry Declerck, Pascal Denis, Tejaswini Deoskar, Mona Diab, Fernando Diaz, Anna Divoli, Bill Dolan, Pinar Donmez, Doug Downey, Markus Dreyer, Rebecca Dridan, Kevin Duh, Chris Dyer, Koji Eguchi, Maud Ehrmann, Jacob Eisenstein, Jason Eisner, Noemie Elhadad, Micha Elsner, Stephanie Elzer, Tomaz Erjavec, Hui Fang, Anna Feldman, Christiane Fellbaum, Denis Filimonov, Jenny Finkel, Dan Flickinger, Radu Florian, Mikel Forcada, Victoria Fossum, George Foster, Jennifer Foster, Alex Fraser, Marjorie Freedman, Maria Fuentes, Pascale Fung, Michel Galley, Michael Gamon, Wei Gao, Yuqing Gao, Claire Gardent, Nikesh Garera, Eric Gaussier, Gary Geunbae Lee, Dafydd Gibbon, Kevin Gimpel, Jesús Giménez, Roxana Girju, Natalie Glance, Randy Goebel, Genevieve Gorrell, Nancy Green, Stephan Greene, Gregory Grefenstette, Ralph Grishman, Jiafeng Guo, Iryna Gurevych, Le An Ha, Ben Hachey, Reza Haffari, Aria Haghighi, Udo Hahn, Dilek Hakkani-Tur, Keith Hall, Kazuo Hara, Sanda Harabagiu, Ahmed Hassan, Hany Hassan, Samer Hassan, Xiaodong He, James Henderson, Julia Hirschberg, Graeme Hirst, Kristy Hollingshead, Tracy Holloway King, Mark Hopkins, Veronique Hoste, Eduard Hovy, Paul Hsu, Pei-Yun Sabrina Hsueh, Yunhua Hu, Chu-Ren Huang, Liang Huang, Minlie Huang, Songfang Huang, Xuanjing Huang, Zhongqiang Huang, Diana Inkpen, Hideki Isozaki, Abe Ittycheriah, Heng Ji, Jing Jiang, Nitin Jindal, Hongyan Jing, Richard Johansson, Mark Johnson, Rie Johnson, Aravind Joshi, Laura Kallmeyer, Lauri Karttunen, Daisuke Kawahara, Junichi Kazama, Genichiro Kikui, Alexandre Klementiev, Kevin Knight, Philipp Koehn, Alek Kolcz, Greg Kondrak, Terry Koo, Moshe Koppel, Anna Korhonen, Zornitsa Kozareva, Geert-Jan Kruijff, Roland Kuhn, Ravi Kumar, Shankar Kumar, Oren Kurland, Sadao Kurohashi, Wai Lam, Guy Lapalme, Matt Lease, Guy Lebanon, James Lester, Lori Levin, Gina-Anne Levow, Mu Li, Shoushan Li, Wenjie Li, Xiao Li, Zhifei Li, Percy Liang, Chin-Yew Lin, Frank Lin, Ken Litkowski, Jingjing Liu, Yan Liu, Yang Liu, Yi Liu, Zhanyi Liu, Adam Lopez, Oier Lopez de Lacalle, Yajuan Lu, Yue Lu, Bin Ma, Yanjun Ma, Bill MacCartney, Craig Macdonald, Wolfgang Macherey, Nitin Madnani, Bernardo Magnini, Suresh Manandhar, Christopher Manning, Daniel Marcu, Mitch Marcus, James Martin, David Martinez, Andre Martins, Yuji Matsumoto, Takuya Matsuzaki, Mausam, Mike Maxwell, Jonathan May, Diana Maynard, Diana McCarthy, David McClosky, Kathy McCoy, Ryan McDonald, Kathy McKeown, Paul McNamee, Michael McTear, Qianzhu Mei, Donald Metzler, Haitao Mi, Tim Miller, Dunja Mladenic, Mary-Francine Moens, Saif Mohammad, Dan Moldovan, Ray Mooney, Alessandro Moschitti, Dragos Munteanu, Gabriel Murray, Masaaki Nagata, Tetsuji Nakagawa, Ramesh Nallapati, Jason Naradowsky, Tahira Naseem, Vivi Nastase, Tetsuya Nasukawa, Roberto Navigli, Ani Nenkova, Hermann Ney, Hwee Tou Ng, Vincent Ng, Jian-Yun Nie, Takashi Ninomiya, Kemal Oflazer, Daisuke Okanohara, Naoaki Okazaki, Manabu Okumura, Miles Osborne, Sebastian Padó, Lluís Padró, Martha Palmer, Sinno Jialin Pan, SooMin Pantel, Ivandre Paraboni, Cecile Paris, Jong Park, Kristen P. Parton, Jon Patrick, Siddharth Patwardhan, Adam Pauls, Fuchun Peng, Gerald Penn, Marco Pennacchiotti, Carol Peters, Slav Petrov, Emily Pitler, Ferran Pla, Massimo Poesio, Joe Polifroni, Simone Paolo Ponzetto, Hoifung Poon, Maja Popovic, John Prager, Stephen Pulman, James Pustejovsky, Sampo Pyysalo, Vahed Qazvinian, Chris Quirk, Stephan Raaijmakers, Ari Rappoport, Manny Rayner, Joseph Reisinger, Philip Resnik, Martin Reynaert,

Reviewers (continued)

Giuseppe Riccardi, Sebastian Riedel, Jason Riesa, Stefan Riezler, German Rigau, Ellen Riloff, Laura Rimell, Christoph Ringlstetter, Alan Ritter, Brian Roark, Mats Rooth, Barbara Rosario, Paolo Rosso, Antti-Veikko Rosti, Kenji Sagae, Horacio Saggion, Rajeev Sangal, Murat Saraclar, Ruhi Sarikaya, Anoop Sarkar, Manabu Sassano, Helmut Schmid, Lane Schwartz, Holger Schwenk, Hendra Setiawan, Burr Settles, Dou Shen, Libin Shen, Wade Shen, Shuming Shi, Luo Si, Advaith Siddharthan, Michel Simard, Khalil Sima'an, Matthew Snover, Ben Snyder, Marina Sokolova, Swapna Somasundaran, Youngin Song, Lucia Specia, Caroline Sporleder, Amanda Stent, Mark Stevenson, Veselin Stoyanov, Michael Strube, Jian Su, Keh-Yih Su, Eiichiro Sumita, Jun Sun, Maosong Sun, Mihai Surdeanu, Charles Sutton, Hisami Suzuki, Krysta Svore, Stan Szpakowicz, Idan Szpektor, Maite Taboada, Hiroya Takamura, Jie Tang, Joel Tetreault, Mariet Theune, Christoph Tillmann, Reut Tsarfaty, Benjamin Tsou, Hajime Tsukada, Dan Tufis, Gokhan Tur, Jordi Turmo, Raghavendra Udupa, Takehito Utsuro, Tim Van de Cruys, Josef van Genabith, Gertjan van Noord, Lucy Vanderwende, Vasudeva Varma, Olga Vechtomova, Ashish Venugopal, Juan Vilar, Andreas Vlachos, Stephen Wan, Xiaojun Wan, Dingding Wang, Haifeng Wang, Wei Wang, Xuanhui Wang, Ye-Yi Wang, Taro Watanabe, Bonnie Webber, Ralph Weischedel, Ji-Rong Wen, Jan Wiebe, Theresa Wilson, Shuly Wintner, Tak-Lam Wong, Dekai Wu, Fei Wu, Hua Wu, Fei Xia, Deyi Xiong, Gu Xu, Jinxi Xu, Peng Xu, Nianwen Xue, Xiaobing Xue, Hirofumi Yamamoto, Kazuhide Yamamoto, Hui Yang, Tao Yang, Roman Yangarber, Alexander Yates, Xing Yi, Scott Yih, Bei Yu, Kun Yu, François Yvon, Jun Zhao, Annie Zaenen, Fabio Massimo Zanzotto, Vanni Zavarella, Jakub Zavrel, Richard Zens, Torsten Zesch, Luke Zettlemoyer, ChengXiang Zhai, Duo Zhang, Hao Zhang, Hui Zhang, Min Zhang, Ying Zhang, Yue Zhang, Guodong Zhou, Jun Zhu, Imed Zitouni, Andreas Zollmann, Chengqing Zong

Secondary Reviewers

Sadaf Abdul-Rauf, Masayuki Asahara, Shen Bin, Yassine Benajiba, Shane Bergsma, Aditya Bhargava, Jiang Bian, Eduardo Blanco, Kristy Elizabeth Boyer, Pere R. Comas, Xiangyu Duan, Aaron Dunlop, Ji Fang, Yi Fang, Wei Gao, Edgar González, Charles Greenbacker, Eun Ha, Stefan Hahn, Zhongjun He, Yun Huang, Adrian Iftene, Tomoya Iwakura, Jiarong Jiang, Bilal Khaliq, Mi-Young Kim, Ioannis Korkontzelos, Sai Krishna, Gregor Leban, Sophia Yat Mei Lee, Shoushan Li, Shuguang Li, Changsong Liu, Shui Liu, Bin Lu, Cynthia Matuszek, Blaz Novak, Antoni Olivé, Elias Ponvert, Rob Phillips, Long Qiu, Longhua Qian, Rodrygo Santos, Kiran Sarvabhotla, Daniel Stein, Qi Su, György Szarvas, David Talbot, Diana Trandabăţ, Emily Tucker Prud'hommeaux, Damiano Spina Valenti, Cristina Vertan, Ying Xu, Shuang-Hong Yang, Limin Yao, Hui Zhang, Lanbo Zhang, Jiajun Zhang

Invited talks

"Why do we call it decoding?"

Kevin Knight, Information Sciences Institute, University of Southern California

The first natural language processing systems had a straightforward goal — decipher coded messages sent by the enemy. Sixty years later, we have many more applications! These include web search, question answering, summarization, speech recognition, and language translation. This talk explores connections between early decipherment research and today's work. We find that many ideas from the earlier era have become core to the field, while others still remain to be picked up and developed.

"Unsupervised feature learning and Deep Learning"

Andrew Ng, Computer Science Department, Stanford University

Machine learning has seen numerous successes, but applying learning algorithms today often means spending a long time laboriously hand-engineering the input feature representation. This is often true for learning in NLP, vision, audio, and many other problems. To address this, recently in machine learning there has been significant interest in unsupervised feature learning algorithms, including "deep learning" algorithms, that can automatically learn rich feature representations from unlabeled data. These algorithms build on such ideas as sparse coding, ICA, and deep belief networks, and have proved very effective for learning good feature representations in many problems. Since these algorithms mostly learn from unlabeled data, they also have the potential to learn from vastly increased amounts of data (since unlabeled data is cheap), and therefore perhaps also achieving vastly improved performance. In this talk, I'll survey the key ideas in this nascent area of unsupervised feature learning and deep learning. I'll outline a few algorithms, and describe a few successful applications of these ideas to problems in NLP, audio/speech, vision, and other problems.

"Challenges in running a commercial search engine"

Amit Singhal, Google, Inc.

These are exciting times for Information Retrieval and NLP. Web search engines have brought IR to the masses. It now affects the lives of hundreds of millions of people, and growing, as Internet search companies launch ever more products based on techniques developed in IR and NLP research.The real world poses unique challenges for search algorithms. They operate at unprecedented scales, and over a wide diversity of information. In addition, we have entered an unprecedented world of "Adversarial Information Retrieval." The lure of billions of dollars of commerce, guided by search engines, motivates all kinds of people to try all kinds of tricks to get their sites to the top of the search results. What techniques do people use to defeat IR algorithms? What are the evaluation challenges for a web search engine? How much impact has IR had on search engines? How does Google serve over 250 Million queries a day, often with sub-second response times? This talk will show that the world of algorithm and system design for commercial search engines can be described by two of Murphy's Laws: a) If anything can go wrong, it will, and b) If anything cannot go wrong, it will anyway.

Fred Jelinek Best Paper Award

The EMNLP-2010 Best Paper Award is named in memory of Fred Jelinek (18 Nov. 1932 - 14 Sept. 2010).

“Dual Decomposition for Parsing with Non-Projective Head Automata,”
Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola and David Sontag

This paper introduces algorithms for non-projective parsing based on *dual decomposition*. We focus on parsing algorithms for *non-projective head automata*, a generalization of head-automata models to non-projective structures. The dual decomposition algorithms are simple and efficient, relying on standard dynamic programming and minimum spanning tree algorithms. They provably solve an LP relaxation of the non-projective parsing problem. Empirically the LP relaxation is very often tight: for many languages, exact solutions are achieved on over 98% of test sentences. The accuracy of our models is higher than previous work on a broad range of datasets.

Best Reviewer Awards

Alex Clark
Amanda Stent
André Martins
Andreas Vlachos
Anja Belz
Anna Korhonen
Aoife Cahill
Beata Beigman Klebanov
Ben Hachey
Bill MacCartney
Bob Damper
Cecile Paris
ChengXiang Zhai
Chin-Yew Lin
Chris Brockett
Chris Cieri
Craig Macdonald
Diana Maynard
Don Metzler
Ellen Riloff
Enrique Alfonseca
Eric Breck
François Yvon
Greg Grefenstette
Greg Kondrak
Helmut Schmid
Ion Androutsopoulos
Jan Wiebe
James Lester
Jason Eisner
John Blitzer
John Prager
Jonathan May
Kathy McCoy
Kemal Oflazer
Kenji Sagae
Kevin Duh
Kristen P. Parton
Laura Kallmeyer
Laura Rimell
Liang Huang
Lucy Vanderwende
Luke Zettlemoyer
Marco Baroni
Maria Fuentes
Mariet Theune
Martin Reynaert
Matt Lease
Maud Ehrmann
Mausam
Michael Cafarella
Michael Collins
Michael Gamon
Michael Strube
Mike Maxwell
Mikel Forcada
Misha Bilenko
Natalie Glance
Nigel Collier
Nitin Madnani
Oren Kurland
Paul Hsu
Paul McNamee
Radu Florian
Raghavendra Udupa
Raman Chandrasekar
Rebecca Dridan
Rie Johnson
Robert Dale
Roland Kuhn
Sabine Bergler
Sampo Pyysalo
Sebastian Riedel
Shuly Wintner
Simone Paolo Ponzetto
Songfang Huang
Stephen Wan
Steve DeNeefe
Swapna Somasundaran
Torsten Zesch
Tracy Holloway King
Veselin Stoyanov
Vincent Ng
Wei Gao
Xiaojun Wan
Zhongqiang Huang

Table of Contents

On Dual Decomposition and Linear Programming Relaxations for Natural Language Processing
Alexander M Rush, David Sontag, Michael Collins and Tommi Jaakkola 1

Self-Training with Products of Latent Variable Grammars
Zhongqiang Huang, Mary Harper and Slav Petrov 12

Utilizing Extra-Sentential Context for Parsing
Jackie Chi Kit Cheung and Gerald Penn 23

Turbo Parsers: Dependency Parsing by Approximate Variational Inference
Andre Martins, Noah Smith, Eric Xing, Pedro Aguiar and Mario Figueiredo 34

Holistic Sentiment Analysis Across Languages: Multilingual Supervised Latent Dirichlet Allocation
Jordan Boyd-Graber and Philip Resnik 45

Jointly Modeling Aspects and Opinions with a MaxEnt-LDA Hybrid
Xin Zhao, Jing Jiang, Hongfei Yan and Xiaoming Li 56

Summarizing Contrastive Viewpoints in Opinionated Text
Michael Paul, ChengXiang Zhai and Roxana Girju 66

Automatically Producing Plot Unit Representations for Narrative Text
Amit Goyal, Ellen Riloff and Hal Daume III 77

Handling Noisy Queries in Cross Language FAQ Retrieval
Danish Contractor, Govind Kothari, Tanveer Faruquie, L V Subramaniam and Sumit Negi 87

Learning the Relative Usefulness of Questions in Community QA
Razvan Bunescu and Yunfeng Huang 97

Positional Language Models for Clinical Information Retrieval
Florian Boudin, Jian-Yun Nie and Martin Dawes 108

Inducing Word Senses to Improve Web Search Result Clustering
Roberto Navigli and Giuseppe Crisafulli 116

Improving Translation via Targeted Paraphrasing
Philip Resnik, Olivia Buzek, Chang Hu, Yakov Kronrod, Alex Quinn and Benjamin B. Bederson
127

Soft Syntactic Constraints for Hierarchical Phrase-Based Translation Using Latent Syntactic Distributions
Zhongqiang Huang, Martin Cmejrek and Bowen Zhou 138

A Hybrid Morpheme-Word Representation for Machine Translation of Morphologically Rich Languages
Minh-Thang Luong, Preslav Nakov and Min-Yen Kan 148

”Poetic” Statistical Machine Translation: Rhyme and Meter
Dmitriy Genzel, Jakob Uszkoreit and Franz Och 158

Efficient Graph-Based Semi-Supervised Learning of Structured Tagging Models
Amarnag Subramanya, Slav Petrov and Fernando Pereira 167

Better Punctuation Prediction with Dynamic Conditional Random Fields
Wei Lu and Hwee Tou Ng 177

Joint Training and Decoding Using Virtual Nodes for Cascaded Segmentation and Tagging Tasks
Xian Qian, Qi Zhang, Yaqian Zhou, Xuanjing Huang and Lide Wu 187

Crouching Dirichlet, Hidden Markov Model: Unsupervised POS Tagging with Context Local Tag Generation
Taesun Moon, Katrin Erk and Jason Baldridge 196

Improving Gender Classification of Blog Authors
Arjun Mukherjee and Bing Liu 207

Negative Training Data Can be Harmful to Text Classification
Xiao-Li Li, Bing Liu and See-Kiong Ng 218

Modeling Organization in Student Essays
Isaac Persing, Alan Davis and Vincent Ng 229

Evaluating Models of Latent Document Semantics in the Presence of OCR Errors
Daniel Walker, William B. Lund and Eric K. Ringger 240

Translingual Document Representations from Discriminative Projections
John Platt, Kristina Toutanova and Wen-tau Yih 251

Storing the Web in Memory: Space Efficient Language Models with Constant Time Retrieval
David Guthrie and Mark Hepple 262

Efficient Incremental Decoding for Tree-to-String Translation
Liang Huang and Haitao Mi 273

Modeling Perspective Using Adaptor Grammars
Eric Hardisty, Jordan Boyd-Graber and Philip Resnik 284

Predicting the Semantic Compositionality of Prefix Verbs
Shane Bergsma, Aditya Bhargava, Hua He and Grzegorz Kondrak 293

Joint Inference for Bilingual Semantic Role Labeling
Tao Zhuang and Chengqing Zong 304

Automatic Discovery of Manner Relations and its Applications
Eduardo Blanco and Dan Moldovan 315

Tense Sense Disambiguation: A New Syntactic Polysemy Task
Roi Reichart and Ari Rappoport . . . 325

Improving Mention Detection Robustness to Noisy Input
Radu Florian, John Pitrelli, Salim Roukos and Imed Zitouni . . . 335

Clustering-Based Stratified Seed Sampling for Semi-Supervised Relation Classification
Longhua Qian and Guodong Zhou . . . 346

Unsupervised Discovery of Negative Categories in Lexicon Bootstrapping
Tara McIntosh . . . 356

Automatic Keyphrase Extraction via Topic Decomposition
Zhiyuan Liu, Wenyi Huang, Yabin Zheng and Maosong Sun . . . 366

Incorporating Content Structure into Text Analysis Applications
Christina Sauper, Aria Haghighi and Regina Barzilay . . . 377

Exploiting Conversation Structure in Unsupervised Topic Segmentation for Emails
Shafiq Joty, Giuseppe Carenini, Gabriel Murray and Raymond T. Ng . . . 388

A Semi-Supervised Approach to Improve Classification of Infrequent Discourse Relations Using Feature Vector Extension
Hugo Hernault, Danushka Bollegala and Mitsuru Ishizuka . . . 399

A Game-Theoretic Approach to Generating Spatial Descriptions
Dave Golland, Percy Liang and Dan Klein . . . 410

Facilitating Translation Using Source Language Paraphrase Lattices
Jinhua Du, Jie Jiang and Andy Way . . . 420

Mining Name Translations from Entity Graph Mapping
Gae-won You, Seung-won Hwang, Young-In Song, Long Jiang and Zaiqing Nie . . . 430

Non-Isomorphic Forest Pair Translation
Hui Zhang, Min Zhang, Haizhou Li and Eng Siong Chng . . . 440

Discriminative Instance Weighting for Domain Adaptation in Statistical Machine Translation
George Foster, Cyril Goutte and Roland Kuhn . . . 451

NLP on Spoken Documents Without ASR
Mark Dredze, Aren Jansen, Glen Coppersmith and Ken Church . . . 460

Fusing Eye Gaze with Speech Recognition Hypotheses to Resolve Exophoric References in Situated Dialogue
Zahar Prasov and Joyce Y. Chai . . . 471

Multi-Document Summarization Using A Search and Discriminative Learning*
Ahmet Aker, Trevor Cohn and Robert Gaizauskas . . . 482

A Multi-Pass Sieve for Coreference Resolution
Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nate Chambers, Mihai Surdeanu, Dan Jurafsky and Christopher Manning 492

A Simple Domain-Independent Probabilistic Approach to Generation
Gabor Angeli, Percy Liang and Dan Klein 502

Title Generation with Quasi-Synchronous Grammar
Kristian Woodsend, Yansong Feng and Mirella Lapata 513

Automatic Analysis of Rhythmic Poetry with Applications to Generation and Translation
Erica Greene, Tugba Bodrumlu and Kevin Knight 524

Discriminative Word Alignment with a Function Word Reordering Model
Hendra Setiawan, Chris Dyer and Philip Resnik 534

Hierarchical Phrase-Based Translation Grammars Extracted from Alignment Posterior Probabilities
Adrià de Gispert, Juan Pino and William Byrne 545

Maximum Entropy Based Phrase Reordering for Hierarchical Phrase-Based Translation
Zhongjun He, Yao Meng and Hao Yu 555

Further Meta-Evaluation of Broad-Coverage Surface Realization
Dominic Espinosa, Rajakrishnan Rajkumar, Michael White and Shoshana Berleant 564

Two Decades of Unsupervised POS Induction: How Far Have We Come?
Christos Christodoulopoulos, Sharon Goldwater and Mark Steedman 575

We're Not in Kansas Anymore: Detecting Domain Changes in Streams
Mark Dredze, Tim Oates and Christine Piatko 585

A Fast Fertility Hidden Markov Model for Word Alignment Using MCMC
Shaojun Zhao and Daniel Gildea 596

Minimum Error Rate Training by Sampling the Translation Lattice
Samidh Chatterjee and Nicola Cancedda 606

Statistical Machine Translation with a Factorized Grammar
Libin Shen, Bing Zhang, Spyros Matsoukas, Jinxi Xu and Ralph Weischedel 616

Discriminative Sample Selection for Statistical Machine Translation
Sankaranarayanan Ananthakrishnan, Rohit Prasad, David Stallard and Prem Natarajan 626

Effects of Empty Categories on Machine Translation
Tagyoung Chung and Daniel Gildea 636

SCFG Decoding Without Binarization
Mark Hopkins and Greg Langmead 646

Example-Based Paraphrasing for Improved Phrase-Based Statistical Machine Translation
Aurélien Max 656

Combining Unsupervised and Supervised Alignments for MT: An Empirical Study
Jinxi Xu and Antti-Veikko Rosti 667

Top-Down Nearly-Context-Sensitive Parsing
Eugene Charniak 674

Improved Fully Unsupervised Parsing with Zoomed Learning
Roi Reichart and Ari Rappoport 684

Unsupervised Parse Selection for HPSG
Rebecca Dridan and Timothy Baldwin 694

Uptraining for Accurate Deterministic Question Parsing
Slav Petrov, Pi-Chuan Chang, Michael Ringgaard and Hiyan Alshawi 705

A Unified Framework for Scope Learning via Simplified Shallow Semantic Parsing
Qiaoming Zhu, Junhui Li, Hongling Wang and Guodong Zhou 714

A New Approach to Lexical Disambiguation of Arabic Text
Rushin Shah, Paramveer S. Dhillon, Mark Liberman, Dean Foster, Mohamed Maamouri and Lyle Ungar 725

What a Parser Can Learn from a Semantic Role Labeler and Vice Versa
Stephen Boxwell, Dennis Mehay and Chris Brew 736

Word Sense Induction & Disambiguation Using Hierarchical Random Graphs
Ioannis Klapaftis and Suresh Manandhar 745

Towards Conversation Entailment: An Empirical Investigation
Chen Zhang and Joyce Chai 756

The Necessity of Combining Adaptation Methods
Ming-Wei Chang, Michael Connor and Dan Roth 767

Training Continuous Space Language Models: Some Practical Issues
Hai Son Le, Alexandre Allauzen, Guillaume Wisniewski and François Yvon 778

Enhancing Domain Portability of Chinese Segmentation Model Using Chi-Square Statistics and Bootstrapping
Baobao Chang and Dongxu Han 789

Latent-Descriptor Clustering for Unsupervised POS Induction
Michael Lamar, Yariv Maron and Elie Bienenstock 799

A Probabilistic Morphological Analyzer for Syriac
Peter McClanahan, George Busby, Robbie Haertel, Kristian Heal, Deryle Lonsdale, Kevin Seppi and Eric Ringger 810

Lessons Learned in Part-of-Speech Tagging of Conversational Speech
Vladimir Eidelman, Zhongqiang Huang and Mary Harper 821

An Efficient Algorithm for Unsupervised Word Segmentation with Branching Entropy and MDL
Valentin Zhikov, Hiroya Takamura and Manabu Okumura 832

A Fast Decoder for Joint Word Segmentation and POS-Tagging Using a Single Discriminative Model
Yue Zhang and Stephen Clark 843

Simple Type-Level Unsupervised POS Tagging
Yoong Keok Lee, Aria Haghighi and Regina Barzilay 853

Classifying Dialogue Acts in One-on-One Live Chats
Su Nam Kim, Lawrence Cavedon and Timothy Baldwin 862

Resolving Event Noun Phrases to Their Verbal Mentions
Bin Chen, Jian Su and Chew Lim Tan 872

A Tree Kernel-Based Unified Framework for Chinese Zero Anaphora Resolution
Fang Kong and Guodong Zhou 882

Automatic Comma Insertion for Japanese Text Generation
Masaki Murata, Tomohiro Ohno and Shigeki Matsubara 892

Using Unknown Word Techniques to Learn Known Words
Kostadin Cholakov and Gertjan van Noord 902

WikiWars: A New Corpus for Research on Temporal Expressions
Pawel Mazur and Robert Dale 913

PEM: A Paraphrase Evaluation Metric Exploiting Parallel Texts
Chang Liu, Daniel Dahlmeier and Hwee Tou Ng 923

Assessing Phrase-Based Translation Models with Oracle Decoding
Guillaume Wisniewski, Alexandre Allauzen and François Yvon 933

Automatic Evaluation of Translation Quality for Distant Language Pairs
Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh and Hajime Tsukada 944

An Approach of Generating Personalized Views from Normalized Electronic Dictionaries : A Practical Experiment on Arabic Language
Aida Khemakhem, Bilel Gargouri and Abdelmajid Ben Hamadou 953

Generating Confusion Sets for Context-Sensitive Error Correction
Alla Rozovskaya and Dan Roth 961

Confidence in Structured-Prediction Using Confidence-Weighted Models
Avihai Mejer and Koby Crammer 971

Evaluating the Impact of Alternative Dependency Graph Encodings on Solving Event Extraction Tasks
Ekaterina Buyko and Udo Hahn 982

Enhancing Mention Detection Using Projection via Aligned Corpora
Yassine Benajiba and Imed Zitouni 993

Domain Adaptation of Rule-Based Annotators for Named-Entity Recognition Tasks
Laura Chiticariu, Rajasekar Krishnamurthy, Yunyao Li, Frederick Reiss and Shivakumar Vaithyanathan 1002

Collective Cross-Document Relation Extraction Without Labelled Data
Limin Yao, Sebastian Riedel and Andrew McCallum 1013

Automatic Detection and Classification of Social Events
Apoorv Agarwal and Owen Rambow 1024

Extracting Opinion Targets in a Single and Cross-Domain Setting with Conditional Random Fields
Niklas Jakob and Iryna Gurevych 1035

Multi-Level Structured Models for Document-Level Sentiment Classification
Ainur Yessenalina, Yisong Yue and Claire Cardie 1046

Cross Language Text Classification by Model Translation and Semi-Supervised Learning
Lei Shi, Rada Mihalcea and Mingjun Tian 1057

SRL-Based Verb Selection for ESL
Xiaohua Liu, Bo Han, Kuan Li, Stephan Hyeonjun Stiller and Ming Zhou 1068

Context Comparison of Bursty Events in Web Search and Online Media
Yunliang Jiang, Cindy Xide Lin and Qiaozhu Mei 1077

Learning First-Order Horn Clauses from Web Text
Stefan Schoenmackers, Jesse Davis, Oren Etzioni and Daniel Weld 1088

Constraints Based Taxonomic Relation Classification
Quang Do and Dan Roth 1099

A Semi-Supervised Method to Learn and Construct Taxonomies Using the Web
Zornitsa Kozareva and Eduard Hovy 1110

Function-Based Question Classification for General QA
Fan Bu, Xingwei Zhu, Yu Hao and Xiaoyan Zhu 1119

Learning Recurrent Event Queries for Web Search
Ruiqiang Zhang, Yuki Konda, Anlei Dong, Pranam Kolari, Yi Chang and Zhaohui Zheng . . . 1129

Staying Informed: Supervised and Semi-Supervised Multi-View Topical Analysis of Ideological Perspective
Amr Ahmed and Eric Xing 1140

Word-Based Dialect Identification with Georeferenced Rules
Yves Scherrer and Owen Rambow 1151

Measuring Distributional Similarity in Context
Georgiana Dinu and Mirella Lapata 1162

A Mixture Model with Sharing for Lexical Semantics
Joseph Reisinger and Raymond Mooney 1173

Nouns are Vectors, Adjectives are Matrices: Representing Adjective-Noun Constructions in Semantic Space
Marco Baroni and Roberto Zamparelli 1183

Practical Linguistic Steganography Using Contextual Synonym Substitution and Vertex Colour Coding
Ching-Yun Chang and Stephen Clark 1194

Unsupervised Induction of Tree Substitution Grammars for Dependency Parsing
Phil Blunsom and Trevor Cohn 1204

It Depends on the Translation: Unsupervised Dependency Parsing via Word Alignment
Samuel Brody 1214

Inducing Probabilistic CCG Grammars from Logical Form with Higher-Order Unification
Tom Kwiatkowksi, Luke Zettlemoyer, Sharon Goldwater and Mark Steedman 1223

Using Universal Linguistic Knowledge to Guide Grammar Induction
Tahira Naseem, Harr Chen, Regina Barzilay and Mark Johnson 1234

What's with the Attitude? Identifying Sentences with Attitude in Online Discussions
Ahmed Hassan, Vahed Qazvinian and Dragomir Radev 1245

Hashing-Based Approaches to Spelling Correction of Personal Names
Raghavendra Udupa and Shaishav Kumar 1256

Identifying Functional Relations in Web Text
Thomas Lin, Mausam and Oren Etzioni 1266

A Latent Variable Model for Geographic Lexical Variation
Jacob Eisenstein, Brendan O'Connor, Noah A. Smith and Eric P. Xing 1277

Dual Decomposition for Parsing with Non-Projective Head Automata
Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola and David Sontag 1288

Conference Program

Saturday, October 9, 2010

8:50–9:00 Opening Remarks

9:00–10:00 Invited talk: Kevin Knight, “Why do we call it decoding?”
CHAIR: HANG LI

10:00–10:30 Break

Session 1A: Syntactic Parsing and Machine Learning
CHAIR: XAVIER CARRERAS

10:30–10:55 *On Dual Decomposition and Linear Programming Relaxations for Natural Language Processing*
Alexander M Rush, David Sontag, Michael Collins and Tommi Jaakkola

10:55–11:20 *Self-Training with Products of Latent Variable Grammars*
Zhongqiang Huang, Mary Harper and Slav Petrov

11:20–11:45 *Utilizing Extra-Sentential Context for Parsing*
Jackie Chi Kit Cheung and Gerald Penn

11:45–12:10 *Turbo Parsers: Dependency Parsing by Approximate Variational Inference*
Andre Martins, Noah Smith, Eric Xing, Pedro Aguiar and Mario Figueiredo

Session 1B: Sentiment Analysis and Opinion Mining
CHAIR: HANNA WALLACH

10:30–10:55 *Holistic Sentiment Analysis Across Languages: Multilingual Supervised Latent Dirichlet Allocation*
Jordan Boyd-Graber and Philip Resnik

10:55–11:20 *Jointly Modeling Aspects and Opinions with a MaxEnt-LDA Hybrid*
Xin Zhao, Jing Jiang, Hongfei Yan and Xiaoming Li

11:20–11:45 *Summarizing Contrastive Viewpoints in Opinionated Text*
Michael Paul, ChengXiang Zhai and Roxana Girju

Saturday, October 9, 2010 (continued)

11:45–12:10 *Automatically Producing Plot Unit Representations for Narrative Text*
Amit Goyal, Ellen Riloff and Hal Daume III

Session 1C: Information Retrieval and Question Answering
CHAIR: JING JIANG

10:30–10:55 *Handling Noisy Queries in Cross Language FAQ Retrieval*
Danish Contractor, Govind Kothari, Tanveer Faruquie, L V Subramaniam and Sumit Negi

10:55–11:20 *Learning the Relative Usefulness of Questions in Community QA*
Razvan Bunescu and Yunfeng Huang

11:20–11:45 *Positional Language Models for Clinical Information Retrieval*
Florian Boudin, Jian-Yun Nie and Martin Dawes

11:45–12:10 *Inducing Word Senses to Improve Web Search Result Clustering*
Roberto Navigli and Giuseppe Crisafulli

12:10–14:10 Lunch

Session 2A: Machine Translation I
CHAIR: DANIEL MARCU

14:10–14:35 *Improving Translation via Targeted Paraphrasing*
Philip Resnik, Olivia Buzek, Chang Hu, Yakov Kronrod, Alex Quinn and Benjamin B. Bederson

14:35–15:00 *Soft Syntactic Constraints for Hierarchical Phrase-Based Translation Using Latent Syntactic Distributions*
Zhongqiang Huang, Martin Cmejrek and Bowen Zhou

15:00–15:25 *A Hybrid Morpheme-Word Representation for Machine Translation of Morphologically Rich Languages*
Minh-Thang Luong, Preslav Nakov and Min-Yen Kan

15:25–15:50 *”Poetic” Statistical Machine Translation: Rhyme and Meter*
Dmitriy Genzel, Jakob Uszkoreit and Franz Och

Saturday, October 9, 2010 (continued)

Session 2B: Tagging, Chunking and Segmentation
CHAIR: SHARON GOLDWATER

14:10–14:35 *Efficient Graph-Based Semi-Supervised Learning of Structured Tagging Models*
Amarnag Subramanya, Slav Petrov and Fernando Pereira

14:35–15:00 *Better Punctuation Prediction with Dynamic Conditional Random Fields*
Wei Lu and Hwee Tou Ng

15:00–15:25 *Joint Training and Decoding Using Virtual Nodes for Cascaded Segmentation and Tagging Tasks*
Xian Qian, Qi Zhang, Yaqian Zhou, Xuanjing Huang and Lide Wu

15:25–15:50 *Crouching Dirichlet, Hidden Markov Model: Unsupervised POS Tagging with Context Local Tag Generation*
Taesun Moon, Katrin Erk and Jason Baldridge

Session 2C: Text Mining
CHAIR: HAL DAUME III

14:10–14:35 *Improving Gender Classification of Blog Authors*
Arjun Mukherjee and Bing Liu

14:35–15:00 *Negative Training Data Can be Harmful to Text Classification*
Xiao-Li Li, Bing Liu and See-Kiong Ng

15:00–15:25 *Modeling Organization in Student Essays*
Isaac Persing, Alan Davis and Vincent Ng

15:25–15:50 *Evaluating Models of Latent Document Semantics in the Presence of OCR Errors*
Daniel Walker, William B. Lund and Eric K. Ringger

Saturday, October 9, 2010 (continued)

15:50–16:20 Break

Session 3A: Machine Learning for NLP
Chair: Noah Smith

16:20–16:45 *Translingual Document Representations from Discriminative Projections*
John Platt, Kristina Toutanova and Wen-tau Yih

16:45–17:10 *Storing the Web in Memory: Space Efficient Language Models with Constant Time Retrieval*
David Guthrie and Mark Hepple

17:10–17:35 *Efficient Incremental Decoding for Tree-to-String Translation*
Liang Huang and Haitao Mi

17:35–18:00 *Modeling Perspective Using Adaptor Grammars*
Eric Hardisty, Jordan Boyd-Graber and Philip Resnik

Session 3B: Semantics
Chair: Mark Steedman

16:20–16:45 *Predicting the Semantic Compositionality of Prefix Verbs*
Shane Bergsma, Aditya Bhargava, Hua He and Grzegorz Kondrak

16:45–17:10 *Joint Inference for Bilingual Semantic Role Labeling*
Tao Zhuang and Chengqing Zong

17:10–17:35 *Automatic Discovery of Manner Relations and its Applications*
Eduardo Blanco and Dan Moldovan

17:35–18:00 *Tense Sense Disambiguation: A New Syntactic Polysemy Task*
Roi Reichart and Ari Rappoport

Saturday, October 9, 2010 (continued)

Session 3C: Information Extraction
Chair: Ellen Riloff

16:20–16:45 *Improving Mention Detection Robustness to Noisy Input*
Radu Florian, John Pitrelli, Salim Roukos and Imed Zitouni

16:45–17:10 *Clustering-Based Stratified Seed Sampling for Semi-Supervised Relation Classification*
Longhua Qian and Guodong Zhou

17:10–17:35 *Unsupervised Discovery of Negative Categories in Lexicon Bootstrapping*
Tara McIntosh

17:35–18:00 *Automatic Keyphrase Extraction via Topic Decomposition*
Zhiyuan Liu, Wenyi Huang, Yabin Zheng and Maosong Sun

Sunday, October 10, 2010

9:00–10:00 Invited talk: Andrew Ng, “Unsupervised feature learning and Deep Learning”
Chair: David Yarowsky

10:00–10:30 Break

Session 4A: Discourse and Dialog
Chair: Rada Mihalcea

10:30–10:55 *Incorporating Content Structure into Text Analysis Applications*
Christina Sauper, Aria Haghighi and Regina Barzilay

10:55–11:20 *Exploiting Conversation Structure in Unsupervised Topic Segmentation for Emails*
Shafiq Joty, Giuseppe Carenini, Gabriel Murray and Raymond T. Ng

11:20–11:45 *A Semi-Supervised Approach to Improve Classification of Infrequent Discourse Relations Using Feature Vector Extension*
Hugo Hernault, Danushka Bollegala and Mitsuru Ishizuka

Sunday, October 10, 2010 (continued)

11:45–12:10 *A Game-Theoretic Approach to Generating Spatial Descriptions*
Dave Golland, Percy Liang and Dan Klein

Session 4B: Machine Translation II
Chair: Chris Quirk

10:30–10:55 *Facilitating Translation Using Source Language Paraphrase Lattices*
Jinhua Du, Jie Jiang and Andy Way

10:55–11:20 *Mining Name Translations from Entity Graph Mapping*
Gae-won You, Seung-won Hwang, Young-In Song, Long Jiang and Zaiqing Nie

11:20–11:45 *Non-Isomorphic Forest Pair Translation*
Hui Zhang, Min Zhang, Haizhou Li and Eng Siong Chng

11:45–12:10 *Discriminative Instance Weighting for Domain Adaptation in Statistical Machine Translation*
George Foster, Cyril Goutte and Roland Kuhn

Session 4C: NLP Applications
Chair: Manabu Okumura

10:30–10:55 *NLP on Spoken Documents Without ASR*
Mark Dredze, Aren Jansen, Glen Coppersmith and Ken Church

10:55–11:20 *Fusing Eye Gaze with Speech Recognition Hypotheses to Resolve Exophoric References in Situated Dialogue*
Zahar Prasov and Joyce Y. Chai

11:20–11:45 *Multi-Document Summarization Using A* Search and Discriminative Learning*
Ahmet Aker, Trevor Cohn and Robert Gaizauskas

11:45–12:10 *A Multi-Pass Sieve for Coreference Resolution*
Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nate Chambers, Mihai Surdeanu, Dan Jurafsky and Christopher Manning

12:10–14:10 Lunch

Session 5A: Natural Language Generation
CHAIR: DRAGOMIR RADEV

14:10–14:35 *A Simple Domain-Independent Probabilistic Approach to Generation*
Gabor Angeli, Percy Liang and Dan Klein

14:35–15:00 *Title Generation with Quasi-Synchronous Grammar*
Kristian Woodsend, Yansong Feng and Mirella Lapata

15:00–15:25 *Automatic Analysis of Rhythmic Poetry with Applications to Generation and Translation*
Erica Greene, Tugba Bodrumlu and Kevin Knight

Session 5B: Machine Translation III
CHAIR: DAVID CHIANG

14:10–14:35 *Discriminative Word Alignment with a Function Word Reordering Model*
Hendra Setiawan, Chris Dyer and Philip Resnik

14:35–15:00 *Hierarchical Phrase-Based Translation Grammars Extracted from Alignment Posterior Probabilities*
Adrià de Gispert, Juan Pino and William Byrne

15:00–15:25 *Maximum Entropy Based Phrase Reordering for Hierarchical Phrase-Based Translation*
Zhongjun He, Yao Meng and Hao Yu

Session 5C: Language Resources
CHAIR: BENJAMIN TSOU

14:10–14:35 *Further Meta-Evaluation of Broad-Coverage Surface Realization*
Dominic Espinosa, Rajakrishnan Rajkumar, Michael White and Shoshana Berleant

14:35–15:00 *Two Decades of Unsupervised POS Induction: How Far Have We Come?*
Christos Christodoulopoulos, Sharon Goldwater and Mark Steedman

15:00–15:25 *We're Not in Kansas Anymore: Detecting Domain Changes in Streams*
Mark Dredze, Tim Oates and Christine Piatko

15:25–15:55 Break

Sunday, October 10, 2010 (continued)

15:55–17:30 **Session 6A: Poster Spotlights**
CHAIR: JASON EISNER

A Fast Fertility Hidden Markov Model for Word Alignment Using MCMC
Shaojun Zhao and Daniel Gildea

Minimum Error Rate Training by Sampling the Translation Lattice
Samidh Chatterjee and Nicola Cancedda

Statistical Machine Translation with a Factorized Grammar
Libin Shen, Bing Zhang, Spyros Matsoukas, Jinxi Xu and Ralph Weischedel

Discriminative Sample Selection for Statistical Machine Translation
Sankaranarayanan Ananthakrishnan, Rohit Prasad, David Stallard and Prem Natarajan

Effects of Empty Categories on Machine Translation
Tagyoung Chung and Daniel Gildea

SCFG Decoding Without Binarization
Mark Hopkins and Greg Langmead

Example-Based Paraphrasing for Improved Phrase-Based Statistical Machine Translation
Aurélien Max

Combining Unsupervised and Supervised Alignments for MT: An Empirical Study
Jinxi Xu and Antti-Veikko Rosti

Top-Down Nearly-Context-Sensitive Parsing
Eugene Charniak

Improved Fully Unsupervised Parsing with Zoomed Learning
Roi Reichart and Ari Rappoport

Unsupervised Parse Selection for HPSG
Rebecca Dridan and Timothy Baldwin

Uptraining for Accurate Deterministic Question Parsing
Slav Petrov, Pi-Chuan Chang, Michael Ringgaard and Hiyan Alshawi

A Unified Framework for Scope Learning via Simplified Shallow Semantic Parsing
Qiaoming Zhu, Junhui Li, Hongling Wang and Guodong Zhou

A New Approach to Lexical Disambiguation of Arabic Text
Rushin Shah, Paramveer S. Dhillon, Mark Liberman, Dean Foster, Mohamed Maamouri and Lyle Ungar

What a Parser Can Learn from a Semantic Role Labeler and Vice Versa
Stephen Boxwell, Dennis Mehay and Chris Brew

Word Sense Induction & Disambiguation Using Hierarchical Random Graphs
Ioannis Klapaftis and Suresh Manandhar

Towards Conversation Entailment: An Empirical Investigation
Chen Zhang and Joyce Chai

The Necessity of Combining Adaptation Methods
Ming-Wei Chang, Michael Connor and Dan Roth

Training Continuous Space Language Models: Some Practical Issues
Hai Son Le, Alexandre Allauzen, Guillaume Wisniewski and François Yvon

15:55–17:25 **Session 6B: Poster Spotlights**
CHAIR: ANDREW MCCALLUM

Enhancing Domain Portability of Chinese Segmentation Model Using Chi-Square Statistics and Bootstrapping
Baobao Chang and Dongxu Han

Latent-Descriptor Clustering for Unsupervised POS Induction
Michael Lamar, Yariv Maron and Elie Bienenstock

A Probabilistic Morphological Analyzer for Syriac
Peter McClanahan, George Busby, Robbie Haertel, Kristian Heal, Deryle Lonsdale, Kevin Seppi and Eric Ringger

Lessons Learned in Part-of-Speech Tagging of Conversational Speech
Vladimir Eidelman, Zhongqiang Huang and Mary Harper

An Efficient Algorithm for Unsupervised Word Segmentation with Branching Entropy and MDL
Valentin Zhikov, Hiroya Takamura and Manabu Okumura

Sunday, October 10, 2010 (continued)

A Fast Decoder for Joint Word Segmentation and POS-Tagging Using a Single Discriminative Model
Yue Zhang and Stephen Clark

Simple Type-Level Unsupervised POS Tagging
Yoong Keok Lee, Aria Haghighi and Regina Barzilay

Classifying Dialogue Acts in One-on-One Live Chats
Su Nam Kim, Lawrence Cavedon and Timothy Baldwin

Resolving Event Noun Phrases to Their Verbal Mentions
Bin Chen, Jian Su and Chew Lim Tan

A Tree Kernel-Based Unified Framework for Chinese Zero Anaphora Resolution
Fang Kong and Guodong Zhou

Automatic Comma Insertion for Japanese Text Generation
Masaki Murata, Tomohiro Ohno and Shigeki Matsubara

Using Unknown Word Techniques to Learn Known Words
Kostadin Cholakov and Gertjan van Noord

WikiWars: A New Corpus for Research on Temporal Expressions
Pawel Mazur and Robert Dale

PEM: A Paraphrase Evaluation Metric Exploiting Parallel Texts
Chang Liu, Daniel Dahlmeier and Hwee Tou Ng

Assessing Phrase-Based Translation Models with Oracle Decoding
Guillaume Wisniewski, Alexandre Allauzen and François Yvon

Automatic Evaluation of Translation Quality for Distant Language Pairs
Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh and Hajime Tsukada

An Approach of Generating Personalized Views from Normalized Electronic Dictionaries : A Practical Experiment on Arabic Language
Aida Khemakhem, Bilel Gargouri and Abdelmajid Ben Hamadou

Generating Confusion Sets for Context-Sensitive Error Correction
Alla Rozovskaya and Dan Roth

15:55–17:25 **Session 6C: Poster Spotlights**
CHAIR: PHILIP RESNIK

Confidence in Structured-Prediction Using Confidence-Weighted Models
Avihai Mejer and Koby Crammer

Evaluating the Impact of Alternative Dependency Graph Encodings on Solving Event Extraction Tasks
Ekaterina Buyko and Udo Hahn

Enhancing Mention Detection Using Projection via Aligned Corpora
Yassine Benajiba and Imed Zitouni

Domain Adaptation of Rule-Based Annotators for Named-Entity Recognition Tasks
Laura Chiticariu, Rajasekar Krishnamurthy, Yunyao Li, Frederick Reiss and Shivakumar Vaithyanathan

Collective Cross-Document Relation Extraction Without Labelled Data
Limin Yao, Sebastian Riedel and Andrew McCallum

Automatic Detection and Classification of Social Events
Apoorv Agarwal and Owen Rambow

Extracting Opinion Targets in a Single and Cross-Domain Setting with Conditional Random Fields
Niklas Jakob and Iryna Gurevych

Multi-Level Structured Models for Document-Level Sentiment Classification
Ainur Yessenalina, Yisong Yue and Claire Cardie

Cross Language Text Classification by Model Translation and Semi-Supervised Learning
Lei Shi, Rada Mihalcea and Mingjun Tian

SRL-Based Verb Selection for ESL
Xiaohua Liu, Bo Han, Kuan Li, Stephan Hyeonjun Stiller and Ming Zhou

Context Comparison of Bursty Events in Web Search and Online Media
Yunliang Jiang, Cindy Xide Lin and Qiaozhu Mei

Sunday, October 10, 2010 (continued)

Learning First-Order Horn Clauses from Web Text
Stefan Schoenmackers, Jesse Davis, Oren Etzioni and Daniel Weld

Constraints Based Taxonomic Relation Classification
Quang Do and Dan Roth

A Semi-Supervised Method to Learn and Construct Taxonomies Using the Web
Zornitsa Kozareva and Eduard Hovy

Function-Based Question Classification for General QA
Fan Bu, Xingwei Zhu, Yu Hao and Xiaoyan Zhu

Learning Recurrent Event Queries for Web Search
Ruiqiang Zhang, Yuki Konda, Anlei Dong, Pranam Kolari, Yi Chang and Zhaohui Zheng

Staying Informed: Supervised and Semi-Supervised Multi-View Topical Analysis of Ideological Perspective
Amr Ahmed and Eric Xing

Word-Based Dialect Identification with Georeferenced Rules
Yves Scherrer and Owen Rambow

17:30–18:00 Break

18:00–21:00 Poster session and Reception

Monday, October 11, 2010

9:00–10:00 Invited talk: Amit Singhal, “Challenges in running a commercial search engine”
CHAIR: KEN CHURCH

10:00–10:30 Break

Session 7A: Lexical Semantics
CHAIR: KRISTINA TOUTANOVA

10:30–10:55 *Measuring Distributional Similarity in Context*
Georgiana Dinu and Mirella Lapata

10:55–11:20 *A Mixture Model with Sharing for Lexical Semantics*
Joseph Reisinger and Raymond Mooney

11:20–11:45 *Nouns are Vectors, Adjectives are Matrices: Representing Adjective-Noun Constructions in Semantic Space*
Marco Baroni and Roberto Zamparelli

11:45–12:10 *Practical Linguistic Steganography Using Contextual Synonym Substitution and Vertex Colour Coding*
Ching-Yun Chang and Stephen Clark

Session 7B: Syntactic Parsing and Grammar Induction
CHAIR: CHRIS MANNING

10:30–10:55 *Unsupervised Induction of Tree Substitution Grammars for Dependency Parsing*
Phil Blunsom and Trevor Cohn

10:55–11:20 *It Depends on the Translation: Unsupervised Dependency Parsing via Word Alignment*
Samuel Brody

11:20–11:45 *Inducing Probabilistic CCG Grammars from Logical Form with Higher-Order Unification*
Tom Kwiatkowksi, Luke Zettlemoyer, Sharon Goldwater and Mark Steedman

11:45–12:10 *Using Universal Linguistic Knowledge to Guide Grammar Induction*
Tahira Naseem, Harr Chen, Regina Barzilay and Mark Johnson

Monday, October 11, 2010 (continued)

Session 7C: NLP for the Web
CHAIR: JIANYUN NIE

10:30–10:55 *What's with the Attitude? Identifying Sentences with Attitude in Online Discussions*
Ahmed Hassan, Vahed Qazvinian and Dragomir Radev

10:55–11:20 *Hashing-Based Approaches to Spelling Correction of Personal Names*
Raghavendra Udupa and Shaishav Kumar

11:20–11:45 *Identifying Functional Relations in Web Text*
Thomas Lin, Mausam, and Oren Etzioni

11:45–12:10 *A Latent Variable Model for Geographic Lexical Variation*
Jacob Eisenstein, Brendan O'Connor, Noah A. Smith and Eric P. Xing

12:15–13:00 SIGDAT Business Meeting

13:00–14:10 Lunch

Plenary Session: Fred Jelinek Best Paper Award
CHAIRS: BOB MOORE, JASON EISNER

14:10–15:05 *Dual Decomposition for Parsing with Non-Projective Head Automata*
Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola and David Sontag

Closing
CHAIR: KEN CHURCH

On Dual Decomposition and Linear Programming Relaxations for Natural Language Processing

Alexander M. Rush **David Sontag** **Michael Collins** **Tommi Jaakkola**
MIT CSAIL, Cambridge, MA 02139, USA
{srush,dsontag,mcollins,tommi}@csail.mit.edu

Abstract

This paper introduces *dual decomposition* as a framework for deriving inference algorithms for NLP problems. The approach relies on standard dynamic-programming algorithms as oracle solvers for sub-problems, together with a simple method for forcing agreement between the different oracles. The approach provably solves a linear programming (LP) relaxation of the global inference problem. It leads to algorithms that are *simple*, in that they use existing decoding algorithms; *efficient*, in that they avoid exact algorithms for the full model; and often *exact*, in that empirically they often recover the correct solution in spite of using an LP relaxation. We give experimental results on two problems: 1) the combination of two lexicalized parsing models; and 2) the combination of a lexicalized parsing model and a trigram part-of-speech tagger.

1 Introduction

Dynamic programming algorithms have been remarkably useful for inference in many NLP problems. Unfortunately, as models become more complex, for example through the addition of new features or components, dynamic programming algorithms can quickly explode in terms of computational or implementational complexity.[1] As a result, efficiency of inference is a critical bottleneck for many problems in statistical NLP.

This paper introduces *dual decomposition* (Dantzig and Wolfe, 1960; Komodakis et al., 2007) as a framework for deriving inference algorithms in NLP. Dual decomposition leverages the observation that complex inference problems can often be decomposed into efficiently solvable sub-problems. The approach leads to inference algorithms with the following properties:

- The resulting algorithms are simple and efficient, building on standard dynamic-programming algorithms as oracle solvers for sub-problems,[2] together with a method for forcing agreement between the oracles.
- The algorithms provably solve a linear programming (LP) relaxation of the original inference problem.
- Empirically, the LP relaxation often leads to an exact solution to the original problem.

The approach is very general, and should be applicable to a wide range of problems in NLP. The connection to linear programming ensures that the algorithms provide a certificate of optimality when they recover the exact solution, and also opens up the possibility of methods that incrementally tighten the LP relaxation until it is exact (Sherali and Adams, 1994; Sontag et al., 2008).

The structure of this paper is as follows. We first give two examples as an illustration of the approach: 1) integrated parsing and trigram part-of-speech (POS) tagging; and 2) combined phrase-structure and dependency parsing. In both settings, it is possible to solve the integrated problem through an "intersected" dynamic program (e.g., for integration of parsing and tagging, the construction from Bar-Hillel et al. (1964) can be used). However, these methods, although polynomial time, are substantially less efficient than our algorithms, and are considerably more complex to implement.

Next, we describe exact polyhedral formulations for the two problems, building on connections between dynamic programming algorithms and marginal polytopes, as described in Martin et al. (1990). These allow us to precisely characterize the relationship between the exact formulations and the

[1] The same is true for NLP inference algorithms based on other exact combinatorial methods, for example methods based on minimum-weight spanning trees (McDonald et al., 2005), or graph cuts (Pang and Lee, 2004).

[2] More generally, other exact inference methods can be used as oracles, for example spanning tree algorithms for non-projective dependency structures.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1–11,
MIT, Massachusetts, USA, 9-11 October 2010.

LP relaxations that we solve. We then give guarantees of convergence for our algorithms by showing that they are instantiations of Lagrangian relaxation, a general method for solving linear programs of a particular form.

Finally, we describe experiments that demonstrate the effectiveness of our approach. First, we consider the integration of the generative model for phrase-structure parsing of Collins (2003), with the second-order discriminative dependency parser of Koo et al. (2008). This is an interesting problem in its own right: the goal is to inject the high performance of discriminative dependency models into phrase-structure parsing. The method uses off-the-shelf decoders for the two models. We find three main results: 1) in spite of solving an LP relaxation, empirically the method finds an exact solution on over 99% of the examples; 2) the method converges quickly, typically requiring fewer than 10 iterations of decoding; 3) the method gives gains over a baseline method that forces the phrase-structure parser to produce the same dependencies as the first-best output from the dependency parser (the Collins (2003) model has an F1 score of 88.1%; the baseline method has an F1 score of 89.7%; and the dual decomposition method has an F1 score of 90.7%).

In a second set of experiments, we use dual decomposition to integrate the trigram POS tagger of Toutanova and Manning (2000) with the parser of Collins (2003). We again find that the method finds an exact solution in almost all cases, with convergence in just a few iterations of decoding.

Although the focus of this paper is on dynamic programming algorithms—both in the experiments, and also in the formal results concerning marginal polytopes—it is straightforward to use other combinatorial algorithms within the approach. For example, Koo et al. (2010) describe a dual decomposition approach for non-projective dependency parsing, which makes use of both dynamic programming and spanning tree inference algorithms.

2 Related Work

Dual decomposition is a classical method for solving optimization problems that can be decomposed into efficiently solvable sub-problems. Our work is inspired by dual decomposition methods for inference in Markov random fields (MRFs) (Wainwright et al., 2005a; Komodakis et al., 2007; Globerson and Jaakkola, 2007). In this approach, the MRF is decomposed into sub-problems corresponding to tree-structured subgraphs that together cover all edges of the original graph. The resulting inference algorithms provably solve an LP relaxation of the MRF inference problem, often significantly faster than commercial LP solvers (Yanover et al., 2006).

Our work is also related to methods that incorporate combinatorial solvers within loopy belief propagation (LBP), either for MAP inference (Duchi et al., 2007) or for computing marginals (Smith and Eisner, 2008). Our approach similarly makes use of combinatorial algorithms to efficiently solve sub-problems of the global inference problem. However, unlike LBP, our algorithms have strong theoretical guarantees, such as guaranteed convergence and the possibility of a certificate of optimality. These guarantees are possible because our algorithms directly solve an LP relaxation.

Other work has considered LP or integer linear programming (ILP) formulations of inference in NLP (Martins et al., 2009; Riedel and Clarke, 2006; Roth and Yih, 2005). These approaches typically use general-purpose LP or ILP solvers. Our method has the advantage that it leverages underlying structure arising in LP formulations of NLP problems. We will see that dynamic programming algorithms such as CKY can be considered to be very efficient solvers for particular LPs. In dual decomposition, these LPs—and their efficient solvers—can be embedded within larger LPs corresponding to more complex inference problems.

3 Background: Structured Models for NLP

We now describe the type of models used throughout the paper. We take some care to set up notation that will allow us to make a clear connection between inference problems and linear programming.

Our first example is weighted CFG parsing. We assume a context-free grammar, in Chomsky normal form, with a set of non-terminals N. The grammar contains all rules of the form $A \rightarrow B\ C$ and $A \rightarrow w$ where $A, B, C \in N$ and $w \in V$ (it is simple to relax this assumption to give a more constrained grammar). For rules of the form $A \rightarrow w$ we refer to A as the part-of-speech tag for w. We allow any non-terminal to be at the root of the tree.

Given a sentence with n words, $w_1, w_2, \ldots w_n$, a parse tree is a set of rule productions of the form $\langle A \rightarrow B\ C, i, k, j\rangle$ where $A, B, C \in N$, and $1 \leq i \leq k < j \leq n$. Each rule production represents the use of CFG rule $A \rightarrow B\ C$ where non-terminal A spans words $w_i \ldots w_j$, non-terminal B spans words $w_i \ldots w_k$, and non-terminal C spans words $w_{k+1} \ldots w_j$. There are $O(|N|^3 n^3)$ such rule productions. Each parse tree corresponds to a subset of these rule productions, of size $n - 1$, that forms a well-formed parse tree.[3]

We now define the *index set* for CFG parsing as

$$\mathcal{I} = \{\langle A \rightarrow B\ C, i, k, j\rangle \colon A, B, C \in N, \\ 1 \leq i \leq k < j \leq n\}$$

Each parse tree is a vector $y = \{y_r : r \in \mathcal{I}\}$, with $y_r = 1$ if rule r is in the parse tree, and $y_r = 0$ otherwise. Hence each parse tree is represented as a vector in $\{0, 1\}^m$, where $m = |\mathcal{I}|$. We use $\mathcal{Y}$ to denote the set of all valid parse-tree vectors; the set $\mathcal{Y}$ is a subset of $\{0, 1\}^m$ (not all binary vectors correspond to valid parse trees).

In addition, we assume a vector $\theta = \{\theta_r : r \in \mathcal{I}\}$ that specifies a weight for each rule production.[4] Each θ_r can take any value in the reals. The optimal parse tree is $y^* = \arg\max_{y \in \mathcal{Y}} \quad y \cdot \theta$ where $y \cdot \theta = \sum_r y_r \theta_r$ is the inner product between y and θ.

We use y_r and $y(r)$ interchangeably (similarly for θ_r and $\theta(r)$) to refer to the r'th component of the vector y. For example $\theta(A \rightarrow B\ C, i, k, j)$ is a weight for the rule $\langle A \rightarrow B\ C, i, k, j\rangle$.

We will use similar notation for other problems. As a second example, in POS tagging the task is to map a sentence of n words $w_1 \ldots w_n$ to a tag sequence $t_1 \ldots t_n$, where each t_i is chosen from a set T of possible tags. We assume a trigram tagger, where a tag sequence is represented through decisions $\langle (A, B) \rightarrow C, i\rangle$ where $A, B, C \in T$, and $i \in \{3 \ldots n\}$. Each production represents a transition where C is the tag of word w_i, and (A, B) are the previous two tags. The index set for tagging is

$$\mathcal{I}_{\text{tag}} = \{\langle (A, B) \rightarrow C, i\rangle \ : \ A, B, C \in T, 3 \leq i \leq n\}$$

Note that we do not need transitions for $i = 1$ or $i = 2$, because the transition $\langle (A, B) \rightarrow C, 3\rangle$ specifies the first three tags in the sentence.[5]

Each tag sequence is represented as a vector $z = \{z_r : r \in \mathcal{I}_{\text{tag}}\}$, and we denote the set of valid tag sequences, a subset of $\{0, 1\}^{|\mathcal{I}_{\text{tag}}|}$, as $\mathcal{Z}$. Given a parameter vector $\theta = \{\theta_r : r \in \mathcal{I}_{\text{tag}}\}$, the optimal tag sequence is $\arg\max_{z \in \mathcal{Z}} z \cdot \theta$.

As a modification to the above approach, we will find it convenient to introduce extended index sets for both the CFG and POS tagging examples. For the CFG case we define the extended index set to be $\mathcal{I}' = \mathcal{I} \cup \mathcal{I}_{\text{uni}}$ where

$$\mathcal{I}_{\text{uni}} = \{(i, t) : i \in \{1 \ldots n\}, t \in T\}$$

Here each pair (i, t) represents word w_i being assigned the tag t. Thus each parse-tree vector y will have additional (binary) components $y(i, t)$ specifying whether or not word i is assigned tag t. (Throughout this paper we will assume that the tag-set used by the tagger, T, is a subset of the set of non-terminals considered by the parser, N.) Note that this representation is over-complete, since a parse tree determines a unique tagging for a sentence: more explicitly, for any $i \in \{1 \ldots n\}$, $Y \in T$, the following linear constraint holds:

$$y(i, Y) = \sum_{k=i+1}^{n} \sum_{X, Z \in N} y(X \rightarrow Y\ Z, i, i, k) + \\ \sum_{k=1}^{i-1} \sum_{X, Z \in N} y(X \rightarrow Z\ Y, k, i-1, i)$$

We apply the same extension to the tagging index set, effectively mapping trigrams down to unigram assignments, again giving an over-complete representation. The extended index set for tagging is referred to as $\mathcal{I}'_{\text{tag}}$.

From here on we will make exclusive use of extended index sets for CFG parsing and trigram tagging. We use the set $\mathcal{Y}$ to refer to the set of valid parse structures under the extended representation;

[3] We do not require rules of the form $A \rightarrow w_i$ in this representation, as they are redundant: specifically, a rule production $\langle A \rightarrow B\ C, i, k, j\rangle$ implies a rule $B \rightarrow w_i$ iff $i = k$, and $C \rightarrow w_j$ iff $j = k + 1$.

[4] We do not require parameters for rules of the form $A \rightarrow w$, as they can be folded into rule production parameters. E.g., under a PCFG we define $\theta(A \rightarrow B\ C, i, k, j) = \log P(A \rightarrow B\ C \mid A) + \delta_{i,k} \log P(B \rightarrow w_i | B) + \delta_{k+1,j} \log P(C \rightarrow w_j | C)$ where $\delta_{x,y} = 1$ if $x = y$, 0 otherwise.

[5] As one example, in an HMM, the parameter $\theta((A, B) \rightarrow C, 3)$ would be $\log P(A | {*}{*}) + \log P(B | {*}A) + \log P(C | AB) + \log P(w_1 | A) + \log P(w_2 | B) + \log P(w_3 | C)$, where $*$ is the start symbol.

each $y \in \mathcal{Y}$ is a binary vector of length $|\mathcal{I}'|$. We similarly use $\mathcal{Z}$ to refer to the set of valid tag structures under the extended representation. We assume parameter vectors for the two problems, $\theta^{\text{cfg}} \in \mathbb{R}^{|\mathcal{I}'|}$ and $\theta^{\text{tag}} \in \mathbb{R}^{|\mathcal{I}'_{\text{tag}}|}$.

4 Two Examples

This section describes the dual decomposition approach for two inference problems in NLP.

4.1 Integrated Parsing and Trigram Tagging

We now describe the dual decomposition approach for integrated parsing and trigram tagging. First, define the set $\mathcal{Q}$ as follows:

$$\mathcal{Q} = \{(y, z) : y \in \mathcal{Y}, z \in \mathcal{Z}, \\ y(i,t) = z(i,t) \text{ for all } (i,t) \in \mathcal{I}_{uni}\} \quad (1)$$

Hence $\mathcal{Q}$ is the set of all (y, z) pairs that agree on their part-of-speech assignments. The integrated parsing and trigram tagging problem is then to solve

$$\max_{(y,z)\in\mathcal{Q}} \left(y \cdot \theta^{\text{cfg}} + z \cdot \theta^{\text{tag}} \right) \quad (2)$$

This problem is equivalent to

$$\max_{y\in\mathcal{Y}} \left(y \cdot \theta^{\text{cfg}} + g(y) \cdot \theta^{\text{tag}} \right)$$

where $g : \mathcal{Y} \rightarrow \mathcal{Z}$ is a function that maps a parse tree y to its set of trigrams $z = g(y)$. The benefit of the formulation in Eq. 2 is that it makes explicit the idea of maximizing over all pairs (y, z) under a set of agreement constraints $y(i,t) = z(i,t)$—this concept will be central to the algorithms in this paper.

With this in mind, we note that we have efficient methods for the inference problems of tagging and parsing alone, and that our combined objective almost separates into these two independent problems. In fact, if we drop the $y(i,t) = z(i,t)$ constraints from the optimization problem, the problem splits into two parts, each of which can be efficiently solved using dynamic programming:

$$(y^*, z^*) = (\arg\max_{y\in\mathcal{Y}} y \cdot \theta^{\text{cfg}}, \arg\max_{z\in\mathcal{Z}} z \cdot \theta^{\text{tag}})$$

Dual decomposition exploits this idea; it results in the algorithm given in figure 1. The algorithm optimizes the combined objective by repeatedly solving the two sub-problems separately—that is, it directly solves the harder optimization problem using an existing CFG parser and trigram tagger. After each iteration the algorithm adjusts the weights $u(i,t)$; these updates modify the objective functions for the two models, encouraging them to agree on the same POS sequence. In section 6.1 we will show that the variables $u(i,t)$ are Lagrange multipliers enforcing agreement constraints, and that the algorithm corresponds to a (sub)gradient method for optimization of a dual function. The algorithm is easy to implement: all that is required is a decoding algorithm for each of the two models, and simple additive updates to the Lagrange multipliers enforcing agreement between the two models.

```
Set $u^{(1)}(i,t) \leftarrow 0$ for all $(i,t) \in \mathcal{I}_{\text{uni}}$
for $k = 1$ to $K$ do
    $y^{(k)} \leftarrow \arg\max_{y\in\mathcal{Y}} \; (y \cdot \theta^{\text{cfg}} - \sum_{(i,t)\in\mathcal{I}_{\text{uni}}} u^{(k)}(i,t)y(i,t))$
    $z^{(k)} \leftarrow \arg\max_{z\in\mathcal{Z}} \; (z \cdot \theta^{\text{tag}} + \sum_{(i,t)\in\mathcal{I}_{\text{uni}}} u^{(k)}(i,t)z(i,t))$
    if $y^{(k)}(i,t) = z^{(k)}(i,t)$ for all $(i,t) \in \mathcal{I}_{\text{uni}}$ then
        return $(y^{(k)}, z^{(k)})$
    for all $(i,t) \in \mathcal{I}_{\text{uni}}$,
        $u^{(k+1)}(i,t) \leftarrow u^{(k)}(i,t) + \alpha_k(y^{(k)}(i,t) - z^{(k)}(i,t))$
return $(y^{(K)}, z^{(K)})$
```

Figure 1: The algorithm for integrated parsing and tagging. The parameters $\alpha_k > 0$ for $k = 1 \ldots K$ specify step sizes for each iteration, and are discussed further in the Appendix. The two $\arg\max$ problems can be solved using dynamic programming.

4.2 Integrating Two Lexicalized Parsers

Our second example problem is the integration of a phrase-structure parser with a higher-order dependency parser. The goal is to add higher-order features to phrase-structure parsing without greatly increasing the complexity of inference.

First, we define an index set for second-order unlabeled projective dependency parsing. The second-order parser considers first-order dependencies, as well as grandparent and sibling second-order dependencies (e.g., see Carreras (2007)). We assume that $\mathcal{I}_{\text{dep}}$ is an index set containing all such dependencies (for brevity we omit the details of this index set). For convenience we define an extended index set that makes explicit use of first-order dependencies, $\mathcal{I}'_{\text{dep}} = \mathcal{I}_{\text{dep}} \cup \mathcal{I}_{\text{first}}$, where

$$\mathcal{I}_{\text{first}} = \{(i,j) \ : \ i \in \{0 \dots n\}, j \in \{1 \dots n\}, i \neq j\}$$

Here (i,j) represents a dependency with head w_i and modifier w_j ($i = 0$ corresponds to the root symbol in the parse). We use $\mathcal{D} \subseteq \{0,1\}^{|\mathcal{I}'_{\text{dep}}|}$ to denote the set of valid projective dependency parses.

The second model we use is a lexicalized CFG. Each symbol in the grammar takes the form $A(h)$ where $A \in N$ is a non-terminal, and $h \in \{1 \dots n\}$ is an index specifying that w_h is the head of the constituent. Rule productions take the form $\langle A(a) \rightarrow B(b)\ C(c), i, k, j\rangle$ where $b \in \{i \dots k\}$, $c \in \{(k+1) \dots j\}$, and a is equal to b or c, depending on whether A receives its head-word from its left or right child. Each such rule implies a dependency (a,b) if $a = c$, or (a,c) if $a = b$. We take $\mathcal{I}_{\text{head}}$ to be the index set of all such rules, and $\mathcal{I}'_{\text{head}} = \mathcal{I}_{\text{head}} \cup \mathcal{I}_{\text{first}}$ to be the extended index set. We define $\mathcal{H} \subseteq \{0,1\}^{|\mathcal{I}'_{\text{head}}|}$ to be the set of valid parse trees.

The integrated parsing problem is then to find

$$(y^*, d^*) = \arg \max_{(y,d) \in \mathcal{R}} \left(y \cdot \theta^{\text{head}} + d \cdot \theta^{\text{dep}} \right) \quad (3)$$

where $\mathcal{R} = \{(y,d) : y \in \mathcal{H}, d \in \mathcal{D},$
$y(i,j) = d(i,j)$ for all $(i,j) \in \mathcal{I}_{\text{first}}\}$

This problem has a very similar structure to the problem of integrated parsing and tagging, and we can derive a similar dual decomposition algorithm. The Lagrange multipliers u are a vector in $\mathbb{R}^{|\mathcal{I}_{\text{first}}|}$ enforcing agreement between dependency assignments. The algorithm (omitted for brevity) is identical to the algorithm in figure 1, but with $\mathcal{I}_{\text{uni}}$, $\mathcal{Y}$, $\mathcal{Z}$, θ^{cfg}, and θ^{tag} replaced with $\mathcal{I}_{\text{first}}$, $\mathcal{H}$, $\mathcal{D}$, θ^{head}, and θ^{dep} respectively. The algorithm only requires decoding algorithms for the two models, together with simple updates to the Lagrange multipliers.

5 Marginal Polytopes and LP Relaxations

We now give formal guarantees for the algorithms in the previous section, showing that they solve LP relaxations of the problems in Eqs. 2 and 3.

To make the connection to linear programming, we first introduce the idea of *marginal polytopes* in section 5.1. In section 5.2, we give a precise statement of the LP relaxations that are being solved by the example algorithms, making direct use of marginal polytopes. In section 6 we will prove that the example algorithms solve these LP relaxations.

5.1 Marginal Polytopes

For a finite set $\mathcal{Y}$, define the set of all distributions over elements in $\mathcal{Y}$ as $\Delta = \{\alpha \in \mathbb{R}^{|\mathcal{Y}|} : \alpha_y \geq 0, \sum_{y \in \mathcal{Y}} \alpha_y = 1\}$. Each $\alpha \in \Delta$ gives a vector of marginals, $\mu = \sum_{y \in \mathcal{Y}} \alpha_y y$, where μ_r can be interpreted as the probability that $y_r = 1$ for a y selected at random from the distribution α.

The set of all possible marginal vectors, known as the *marginal polytope*, is defined as follows:

$$\mathcal{M} = \{\mu \in \mathbb{R}^m : \exists \alpha \in \Delta \text{ such that } \mu = \sum_{y \in \mathcal{Y}} \alpha_y y\}$$

$\mathcal{M}$ is also frequently referred to as the *convex hull* of $\mathcal{Y}$, written as $\text{conv}(\mathcal{Y})$. We use the notation $\text{conv}(\mathcal{Y})$ in the remainder of this paper, instead of $\mathcal{M}$.

For an arbitrary set $\mathcal{Y}$, the marginal polytope $\text{conv}(\mathcal{Y})$ can be complex to describe.[6] However, Martin et al. (1990) show that for a very general class of dynamic programming problems, the corresponding marginal polytope can be expressed as

$$\text{conv}(\mathcal{Y}) = \{\mu \in \mathbb{R}^m : A\mu = b, \mu \geq 0\} \quad (4)$$

where A is a $p \times m$ matrix, b is vector in $\mathbb{R}^p$, and the value p is linear in the size of a hypergraph representation of the dynamic program. Note that A and b specify a set of p linear constraints.

We now give an explicit description of the resulting constraints for CFG parsing:[7] similar constraints arise for other dynamic programming algorithms for parsing, for example the algorithms of Eisner (2000). The exact form of the constraints, and the fact that they are polynomial in number, is not essential for the formal results in this paper. However, a description of the constraints gives valuable intuition for the structure of the marginal polytope.

The constraints are given in figure 2. To develop some intuition, consider the case where the variables μ_r are restricted to be binary: hence each binary vector μ specifies a parse tree. The second constraint in Eq. 5 specifies that exactly one rule must be used at the top of the tree. The set of constraints in Eq. 6 specify that for each production of the form

[6] For any finite set $\mathcal{Y}$, $\text{conv}(\mathcal{Y})$ can be expressed as $\{\mu \in \mathbb{R}^m : A\mu \leq b\}$ where A is a matrix of dimension $p \times m$, and $b \in \mathbb{R}^p$ (see, e.g., Korte and Vygen (2008), pg. 65). The value for p depends on the set $\mathcal{Y}$, and can be exponential in size.

[7] Taskar et al. (2004) describe the same set of constraints, but without proof of correctness or reference to Martin et al. (1990).

$$\forall r \in \mathcal{I}',\ \mu_r \geq 0\ ;\quad \sum_{\substack{X,Y,Z \in N\\ k=1\ldots(n-1)}} \mu(X \rightarrow Y\ Z, 1, k, n) = 1 \tag{5}$$

$\forall X \in N$, $\forall (i,j)$ such that $1 \leq i < j \leq n$ and $(i,j) \neq (1,n)$:

$$\sum_{\substack{Y,Z \in N\\ k=i\ldots(j-1)}} \mu(X \rightarrow Y\ Z, i, k, j) = \sum_{\substack{Y,Z \in N\\ k=1\ldots(i-1)}} \mu(Y \rightarrow Z\ X, k, i-1, j) + \sum_{\substack{Y,Z \in N\\ k=(j+1)\ldots n}} \mu(Y \rightarrow X\ Z, i, j, k) \tag{6}$$

$$\forall Y \in T,\ \forall i \in \{1 \ldots n\}:\quad \mu(i,Y) = \sum_{\substack{X,Z \in N\\ k=(i+1)\ldots n}} \mu(X \rightarrow Y\ Z, i, i, k) + \sum_{\substack{X,Z \in N\\ k=1\ldots(i-1)}} \mu(X \rightarrow Z\ Y, k, i-1, i) \tag{7}$$

Figure 2: The linear constraints defining the marginal polytope for CFG parsing.

$\langle X \rightarrow Y\ Z, i, k, j\rangle$ in a parse tree, there must be exactly one production higher in the tree that generates (X, i, j) as one of its children. The constraints in Eq. 7 enforce consistency between the $\mu(i, Y)$ variables and rule variables higher in the tree. Note that the constraints in Eqs.(5–7) can be written in the form $A\mu = b$, $\mu \geq 0$, as in Eq. 4.

Under these definitions, we have the following:

Theorem 5.1 *Define $\mathcal{Y}$ to be the set of all CFG parses, as defined in section 4. Then*

$$\text{conv}(\mathcal{Y}) = \{\mu \in \mathbb{R}^m : \mu \textit{ satisifies Eqs.(5–7)}\}$$

Proof: This theorem is a special case of Martin et al. (1990), theorem 2.

The marginal polytope for tagging, $\text{conv}(\mathcal{Z})$, can also be expressed using linear constraints as in Eq. 4; see figure 3. These constraints follow from results for graphical models (Wainwright and Jordan, 2008), or from the Martin et al. (1990) construction.

As a final point, the following theorem gives an important property of marginal polytopes, which we will use at several points in this paper:

Theorem 5.2 *(Korte and Vygen (2008), page 66.) For any set $\mathcal{Y} \subseteq \{0,1\}^k$, and for any vector $\theta \in \mathbb{R}^k$,*

$$\max_{y \in \mathcal{Y}} y \cdot \theta = \max_{\mu \in \text{conv}(\mathcal{Y})} \mu \cdot \theta \tag{8}$$

$$\forall r \in \mathcal{I}'_{\text{tag}},\ \nu_r \geq 0\ ;\quad \sum_{X,Y,Z \in T} \nu((X,Y) \rightarrow Z, 3) = 1$$

$\forall X \in T, \forall i \in \{3 \ldots n-1\}$:

$$\sum_{Y,Z \in T} \nu((Y,Z) \rightarrow X, i) = \sum_{Y,Z \in T} \nu((Y,X) \rightarrow Z, i+1)$$

$\forall X \in T, \forall i \in \{3 \ldots n-2\}$:

$$\sum_{Y,Z \in T} \nu((Y,Z) \rightarrow X, i) = \sum_{Y,Z \in T} \nu((X,Y) \rightarrow Z, i+2)$$

$$\forall X \in T, \forall i \in \{3 \ldots n\}:\ \nu(i,X) = \sum_{Y,Z \in T} \nu((Y,Z) \rightarrow X, i)$$

$$\forall X \in T:\quad \nu(1,X) = \sum_{Y,Z \in T} \nu((X,Y) \rightarrow Z, 3)$$

$$\forall X \in T:\quad \nu(2,X) = \sum_{Y,Z \in T} \nu((Y,X) \rightarrow Z, 3)$$

Figure 3: The linear constraints defining the marginal polytope for trigram POS tagging.

The theorem states that for a linear objective function, maximization over a discrete set $\mathcal{Y}$ can be replaced by maximization over the convex hull $\text{conv}(\mathcal{Y})$. The problem $\max_{\mu \in \text{conv}(\mathcal{Y})} \mu \cdot \theta$ is a linear programming problem.

For parsing, this theorem implies that:

1. Weighted CFG parsing can be framed as a linear programming problem, of the form $\max_{\mu \in \text{conv}(\mathcal{Y})} \mu \cdot \theta$, where $\text{conv}(\mathcal{Y})$ is specified by a polynomial number of linear constraints.

2. Conversely, dynamic programming algorithms such as the CKY algorithm can be considered to be oracles that efficiently solve LPs of the form $\max_{\mu \in \text{conv}(\mathcal{Y})} \mu \cdot \theta$.

Similar results apply for the POS tagging case.

5.2 Linear Programming Relaxations

We now describe the LP relaxations that are solved by the example algorithms in section 4. We begin with the algorithm in Figure 1.

The original optimization problem was to find $\max_{(y,z) \in \mathcal{Q}} \left(y \cdot \theta^{\text{cfg}} + z \cdot \theta^{\text{tag}}\right)$ (see Eq. 2). By theorem 5.2, this is equivalent to solving

$$\max_{(\mu,\nu) \in \text{conv}(\mathcal{Q})} \left(\mu \cdot \theta^{\text{cfg}} + \nu \cdot \theta^{\text{tag}}\right) \tag{9}$$

To formulate our approximation, we first define:

$$\mathcal{Q}' = \{(\mu,\nu) : \mu \in \text{conv}(\mathcal{Y}), \nu \in \text{conv}(\mathcal{Z}),\ \mu(i,t) = \nu(i,t) \text{ for all } (i,t) \in \mathcal{I}_{\text{uni}}\}$$

The definition of $\mathcal{Q}'$ is very similar to the definition of $\mathcal{Q}$ (see Eq. 1), the only difference being that $\mathcal{Y}$ and $\mathcal{Z}$ are replaced by $\text{conv}(\mathcal{Y})$ and $\text{conv}(\mathcal{Z})$ respectively. Hence any point in $\mathcal{Q}$ is also in $\mathcal{Q}'$. It follows that any point in $\text{conv}(\mathcal{Q})$ is also in $\mathcal{Q}'$, because $\mathcal{Q}'$ is a convex set defined by linear constraints.

The LP relaxation then corresponds to the following optimization problem:

$$\max_{(\mu,\nu)\in\mathcal{Q}'} \left(\mu \cdot \theta^{\text{cfg}} + \nu \cdot \theta^{\text{tag}}\right) \tag{10}$$

$\mathcal{Q}'$ is defined by linear constraints, making this a linear program. Since $\mathcal{Q}'$ is an *outer bound* on $\text{conv}(\mathcal{Q})$, i.e. $\text{conv}(\mathcal{Q}) \subseteq \mathcal{Q}'$, we obtain the guarantee that the value of Eq. 10 always upper bounds the value of Eq. 9.

In Appendix A we give an example showing that in general $\mathcal{Q}'$ includes points that are not in $\text{conv}(\mathcal{Q})$. These points exist because the agreement between the two parts is now enforced in expectation ($\mu(i,t) = \nu(i,t)$ for $(i,t) \in \mathcal{I}_{\text{uni}}$) rather than based on actual assignments. This agreement constraint is weaker since different distributions over assignments can still result in the same first order expectations. Thus, the solution to Eq. 10 may be in $\mathcal{Q}'$ but not in $\text{conv}(\mathcal{Q})$. It can be shown that all such solutions will be *fractional*, making them easy to distinguish from $\mathcal{Q}$. In many applications of LP relaxations—including the examples discussed in this paper—the relaxation in Eq. 10 turns out to be *tight*, in that the solution is often integral (i.e., it is in $\mathcal{Q}$). In these cases, solving the LP relaxation *exactly* solves the original problem of interest.

In the next section we prove that the algorithm in Figure 1 solves the problem in Eq 10. A similar result holds for the algorithm in section 4.2: it solves a relaxation of Eq. 3, where $\mathcal{R}$ is replaced by

$$\mathcal{R}' = \{(\mu,\nu) : \mu \in \text{conv}(\mathcal{H}), \nu \in \text{conv}(\mathcal{D}),$$
$$\mu(i,j) = \nu(i,j) \text{ for all } (i,j) \in \mathcal{I}_{\text{first}}\}$$

6 Convergence Guarantees

6.1 Lagrangian Relaxation

We now show that the example algorithms solve their respective LP relaxations given in the previous section. We do this by first introducing a general class of linear programs, together with an optimization method, *Lagrangian relaxation*, for solving these LPs. We then show that the algorithms in section 4 are special cases of the general algorithm.

The linear programs we consider take the form

$$\max_{x_1\in X_1, x_2\in X_2} (\theta_1 \cdot x_1 + \theta_2 \cdot x_2) \quad \text{such that } Ex_1 = Fx_2$$

The matrices $E \in \mathbb{R}^{q\times m}$ and $F \in \mathbb{R}^{q\times l}$ specify q linear "agreement" constraints between $x_1 \in \mathbb{R}^m$ and $x_2 \in \mathbb{R}^l$. The sets X_1, X_2 are also specified by linear constraints, $X_1 = \{x_1 \in \mathbb{R}^m : Ax_1 = b, x_1 \geq 0\}$ and $X_2 = \left\{x_2 \in \mathbb{R}^l : Cx_2 = d, x_2 \geq 0\right\}$, hence the problem is an LP.

Note that if we set $X_1 = \text{conv}(\mathcal{Y})$, $X_2 = \text{conv}(\mathcal{Z})$, and define E and F to specify the agreement constraints $\mu(i,t) = \nu(i,t)$, then we have the LP relaxation in Eq. 10.

It is natural to apply Lagrangian relaxation in cases where the sub-problems $\max_{x_1\in X_1} \theta_1 \cdot x_1$ and $\max_{x_2\in X_2} \theta_2 \cdot x_2$ can be efficiently solved by combinatorial algorithms for any values of θ_1, θ_2, but where the constraints $Ex_1 = Fx_2$ "complicate" the problem. We introduce Lagrange multipliers $u \in \mathbb{R}^q$ that enforce the latter set of constraints, giving the Lagrangian:

$$L(u, x_1, x_2) = \theta_1 \cdot x_1 + \theta_2 \cdot x_2 + u \cdot (Ex_1 - Fx_2)$$

The dual objective function is

$$L(u) = \max_{x_1\in X_1, x_2\in X_2} L(u, x_1, x_2)$$

and the dual problem is to find $\min_{u\in\mathbb{R}^q} L(u)$.

Because X_1 and X_2 are defined by linear constraints, by strong duality we have

$$\min_{u\in\mathbb{R}^q} L(u) = \max_{x_1\in X_1, x_2\in X_2 : Ex_1=Fx_2} (\theta_1 \cdot x_1 + \theta_2 \cdot x_2)$$

Hence minimizing $L(u)$ will recover the maximum value of the original problem. This leaves open the question of how to recover the LP solution (i.e., the pair (x_1^*, x_2^*) that achieves this maximum); we discuss this point in section 6.2.

The dual $L(u)$ is convex. However, $L(u)$ is not differentiable, so we cannot use gradient-based methods to optimize it. Instead, a standard approach is to use a subgradient method. Subgradients are tangent lines that lower bound a function even at points of non-differentiability: formally, a subgradient of a convex function $L : \mathcal{R}^n \to \mathcal{R}$ at a point u is a vector g_u such that for all v, $L(v) \geq L(u) + g_u \cdot (v - u)$.

```
$u^{(1)} \leftarrow 0$
**for** $k = 1$ to $K$ **do**
    $x_1^{(k)} \leftarrow \arg\max_{x_1 \in X_1} (\theta_1 + (u^{(k)})^T E) \cdot x_1$
    $x_2^{(k)} \leftarrow \arg\max_{x_2 \in X_2} (\theta_2 - (u^{(k)})^T F) \cdot x_2$
    **if** $Ex_1^{(k)} = Fx_2^{(k)}$ **return** $u^{(k)}$
    $u^{(k+1)} \leftarrow u^{(k)} - \alpha_k (Ex_1^{(k)} - Fx_2^{(k)})$
**return** $u^{(K)}$
```

Figure 4: The Lagrangian relaxation algorithm.

By standard results, the subgradient for L at a point u takes a simple form, $g_u = Ex_1^* - Fx_2^*$, where

$$x_1^* = \arg \max_{x_1 \in X_1} (\theta_1 + (u^{(k)})^T E) \cdot x_1$$

$$x_2^* = \arg \max_{x_2 \in X_2} (\theta_2 - (u^{(k)})^T F) \cdot x_2$$

The beauty of this result is that the values of x_1^* and x_2^*, and by implication the value of the subgradient, can be computed using oracles for the two $\arg\max$ sub-problems.

Subgradient algorithms perform updates that are similar to gradient descent:

$$u^{(k+1)} \leftarrow u^{(k)} - \alpha_k g^{(k)}$$

where $g^{(k)}$ is the subgradient of L at $u^{(k)}$ and $\alpha_k > 0$ is the step size of the update. The complete subgradient algorithm is given in figure 4. The following convergence theorem is well-known (e.g., see page 120 of Korte and Vygen (2008)):

Theorem 6.1 *If* $\lim_{k \to \infty} \alpha_k = 0$ *and* $\sum_{k=1}^{\infty} \alpha_k = \infty$*, then* $\lim_{k \to \infty} L(u^{(k)}) = \min_u L(u)$.

The following proposition is easily verified:

Proposition 6.1 *The algorithm in figure 1 is an instantiation of the algorithm in figure 4,*[8] *with* $X_1 = \text{conv}(\mathcal{Y})$, $X_2 = \text{conv}(\mathcal{Z})$*, and the matrices* E *and* F *defined to be binary matrices specifying the constraints* $\mu(i,t) = \nu(i,t)$ *for all* $(i,t) \in \mathcal{I}_{uni}$.

Under an appropriate definition of the step sizes α_k, it follows that the algorithm in figure 1 defines a sequence of Lagrange multipliers $u^{(k)}$ minimizing a dual of the LP relaxation in Eq. 10. A similar result holds for the algorithm in section 4.2.

6.2 Recovering the LP Solution

The previous section described how the method in figure 4 can be used to minimize the dual $L(u)$ of the original linear program. We now turn to the problem of recovering a primal solution (x_1^*, x_2^*) of the LP. The method we propose considers two cases:

(Case 1) If $Ex_1^{(k)} = Fx_2^{(k)}$ at any stage during the algorithm, then simply take $(x_1^{(k)}, x_2^{(k)})$ to be the primal solution. In this case the pair $(x_1^{(k)}, x_2^{(k)})$ *exactly* solves the original LP.[9] If this case arises in the algorithm in figure 1, then the resulting solution is binary (i.e., it is a member of $\mathcal{Q}$), and the solution exactly solves the original inference problem.

(Case 2) If case 1 does not arise, then a couple of strategies are possible. (This situation could arise in cases where the LP is not tight—i.e., it has a fractional solution—or where K is not large enough for convergence.) The first is to define the primal solution to be the average of the solutions encountered during the algorithm: $\hat{x}_1 = \sum_k x_1^{(k)}/K$, $\hat{x}_2 = \sum_k x_2^{(k)}/K$. Results from Nedić and Ozdaglar (2009) show that as $K \to \infty$, these averaged solutions converge to the optimal primal solution.[10] A second strategy (as given in figure 1) is to simply take $(x_1^{(K)}, x_2^{(K)})$ as an approximation to the primal solution. This method is a heuristic, but previous work (e.g., Komodakis et al. (2007)) has shown that it is effective in practice; we use it in this paper.

In our experiments we found that in the vast majority of cases, case 1 applies, after a small number of iterations; see the next section for more details.

7 Experiments

7.1 Integrated Phrase-Structure and Dependency Parsing

Our first set of experiments considers the integration of Model 1 of Collins (2003) (a lexicalized phrase-structure parser, from here on referred to as Model

[8] with the caveat that it returns $(x_1^{(k)}, x_2^{(k)})$ rather than $u^{(k)}$.

[9] We have that $\theta_1 \cdot x_1^{(k)} + \theta_2 \cdot x_2^{(k)} = L(u^{(k)}, x_1^{(k)}, x_2^{(k)}) = L(u^{(k)})$, where the last equality is because $x_1^{(k)}$ and $x_2^{(k)}$ are defined by the respective $\arg\max$'s. Thus, $(x_1^{(k)}, x_2^{(k)})$ and $u^{(k)}$ are primal and dual optimal.

[10] The resulting fractional solution can be projected back to the set $\mathcal{Q}$, see (Smith and Eisner, 2008; Martins et al., 2009).

Itn.	1	2	3	4	5-10	11-20	20-50	**
Dep	43.5	20.1	10.2	4.9	14.0	5.7	1.4	0.4
POS	58.7	15.4	6.3	3.6	10.3	3.8	0.8	1.1

Table 1: Convergence results for Section 23 of the WSJ Treebank for the dependency parsing and POS experiments. Each column gives the percentage of sentences whose *exact* solutions were found in a given range of subgradient iterations. ** is the percentage of sentences that did not converge by the iteration limit (K=50).

1),[11] and the 2nd order discriminative dependency parser of Koo et al. (2008). The inference problem for a sentence x is to find

$$y^* = \arg\max_{y \in \mathcal{Y}} (f_1(y) + \gamma f_2(y)) \qquad (11)$$

where $\mathcal{Y}$ is the set of all lexicalized phrase-structure trees for the sentence x; $f_1(y)$ is the score (log probability) under Model 1; $f_2(y)$ is the score under Koo et al. (2008) for the dependency structure implied by y; and $\gamma > 0$ is a parameter dictating the relative weight of the two models.[12] This problem is similar to the second example in section 4; a very similar dual decomposition algorithm to that described in section 4.2 can be derived.

We used the Penn Wall Street Treebank (Marcus et al., 1994) for the experiments, with sections 2-21 for training, section 22 for development, and section 23 for testing. The parameter γ was chosen to optimize performance on the development set.

We ran the dual decomposition algorithm with a limit of $K = 50$ iterations. The dual decomposition algorithm returns an exact solution if case 1 occurs as defined in section 6.2; we found that of 2416 sentences in section 23, case 1 occurred for 2407 (99.6%) sentences. Table 1 gives statistics showing the number of iterations required for convergence. Over 80% of the examples converge in 5 iterations or fewer; over 90% converge in 10 iterations or fewer.

	Precision	Recall	F$_1$	Dep
Model 1	88.4	87.8	88.1	91.4
Koo08 Baseline	89.9	89.6	89.7	93.3
DD Combination	91.0	90.4	90.7	93.8

Table 2: Performance results for Section 23 of the WSJ Treebank. Model 1: a reimplementation of the generative parser of (Collins, 2002). Koo08 Baseline: Model 1 with a hard restriction to dependencies predicted by the discriminative dependency parser of (Koo et al., 2008). DD Combination: a model that maximizes the joint score of the two parsers. Dep shows the unlabeled dependency accuracy of each system.

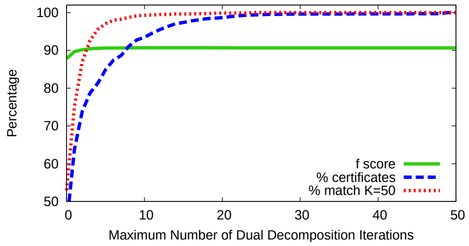


Figure 5: Performance on the parsing task assuming a fixed number of iterations K. f-score: accuracy of the method. % certificates: percentage of examples for which a certificate of optimality is provided. % match: percentage of cases where the output from the method is identical to the output when using $K = 50$.

We compare the accuracy of the dual decomposition approach to two baselines: first, Model 1; and second, a naive integration method that enforces the hard constraint that Model 1 must only consider dependencies seen in the first-best output from the dependency parser. Table 2 shows all three results. The dual decomposition method gives a significant gain in precision and recall over the naive combination method, and boosts the performance of Model 1 to a level that is close to some of the best single-pass parsers on the Penn treebank test set. Dependency accuracy is also improved over the Koo et al. (2008) model, in spite of the relatively low dependency accuracy of Model 1 alone.

Figure 5 shows performance of the approach as a function of K, the maximum number of iterations of dual decomposition. For this experiment, for cases where the method has not converged for $k \leq K$, the output from the algorithm is chosen to be the $y^{(k)}$ for $k \leq K$ that maximizes the objective function in Eq. 11. The graphs show that values of K less than 50 produce almost identical performance to $K = 50$, but with fewer cases giving certificates of optimality (with $K = 10$, the f-score of the method is 90.69%; with $K = 5$ it is 90.63%).

[11] We use a reimplementation that is a slight modification of Collins Model 1, with very similar performance, and which uses the TAG formalism of Carreras et al. (2008).

[12] Note that the models f_1 and f_2 were trained separately, using the methods described by Collins (2003) and Koo et al. (2008) respectively.

	Precision	Recall	F_1	POS Acc
Fixed Tags	88.1	87.6	87.9	96.7
DD Combination	88.7	88.0	88.3	97.1

Table 3: Performance results for Section 23 of the WSJ. Model 1 (Fixed Tags): a baseline parser initialized to the best tag sequence of from the tagger of Toutanova and Manning (2000). DD Combination: a model that maximizes the joint score of parse and tag selection.

7.2 Integrated Phrase-Structure Parsing and Trigram POS tagging

In a second experiment, we used dual decomposition to integrate the Model 1 parser with the Stanford max-ent trigram POS tagger (Toutanova and Manning, 2000), using a very similar algorithm to that described in section 4.1. We use the same training/dev/test split as in section 7.1. The two models were again trained separately.

We ran the algorithm with a limit of $K = 50$ iterations. Out of 2416 test examples, the algorithm found an exact solution in 98.9% of the cases. Table 1 gives statistics showing the speed of convergence for different examples: over 94% of the examples converge to an exact solution in 10 iterations or fewer. In terms of accuracy, we compare to a baseline approach of using the first-best tag sequence as input to the parser. The dual decomposition approach gives 88.3 F1 measure in recovering parse-tree constituents, compared to 87.9 for the baseline.

8 Conclusions

We have introduced dual-decomposition algorithms for inference in NLP, given formal properties of the algorithms in terms of LP relaxations, and demonstrated their effectiveness on problems that would traditionally be solved using intersections of dynamic programs (Bar-Hillel et al., 1964). Given the widespread use of dynamic programming in NLP, there should be many applications for the approach.

There are several possible extensions of the method we have described. We have focused on cases where two models are being combined; the extension to more than two models is straightforward (e.g., see Komodakis et al. (2007)). This paper has considered approaches for MAP inference; for closely related methods that compute approximate marginals, see Wainwright et al. (2005b).

A Fractional Solutions

We now give an example of a point $(\mu, \nu) \in \mathcal{Q}' \backslash \text{conv}(\mathcal{Q})$ that demonstrates that the relaxation $\mathcal{Q}'$ is strictly larger than $\text{conv}(\mathcal{Q})$. Fractional points such as this one can arise as solutions of the LP relaxation for worst case instances, preventing us from finding an exact solution.

Recall that the constraints for $\mathcal{Q}'$ specify that $\mu \in \text{conv}(\mathcal{Y})$, $\nu \in \text{conv}(\mathcal{Z})$, and $\mu(i,t) = \nu(i,t)$ for all $(i,t) \in \mathcal{I}_{\text{uni}}$. Since $\mu \in \text{conv}(\mathcal{Y})$, μ must be a convex combination of 1 or more members of $\mathcal{Y}$; a similar property holds for ν. The example is as follows. There are two possible parts of speech, A and B, and an additional non-terminal symbol X. The sentence is of length 3, $w_1\ w_2\ w_3$. Let ν be the convex combination of the following two tag sequences, each with probability 0.5: $w_1/A\ w_2/A\ w_3/A$ and $w_1/A\ w_2/B\ w_3/B$. Let μ be the convex combination of the following two parses, each with probability 0.5: $(X(A\ w_1)(X(A\ w_2)(B\ w_3)))$ and $(X(A\ w_1)(X(B\ w_2)(A\ w_3)))$. It can be verified that $\mu(i,t) = \nu(i,t)$ for all (i,t), i.e., the marginals for single tags for μ and ν agree. Thus, $(\mu,\nu) \in \mathcal{Q}'$.

To demonstrate that this fractional point is *not* in $\text{conv}(\mathcal{Q})$, we give parameter values such that this fractional point is optimal and all integral points (i.e., actual parses) are suboptimal. For the tagging model, set $\theta(AA \rightarrow A, 3) = \theta(AB \rightarrow B, 3) = 0$, with all other parameters having a negative value. For the parsing model, set $\theta(X \rightarrow A\ X, 1, 1, 3) = \theta(X \rightarrow A\ B, 2, 2, 3) = \theta(X \rightarrow B\ A, 2, 2, 3) = 0$, with all other rule parameters being negative. For this objective, the fractional solution has value 0, while all integral points (i.e., all points in $\mathcal{Q}$) have a negative value. By Theorem 5.2, the maximum of any linear objective over $\text{conv}(\mathcal{Q})$ is equal to the maximum over $\mathcal{Q}$. Thus, $(\mu,\nu) \notin \text{conv}(\mathcal{Q})$.

B Step Size

We used the following step size in our experiments. First, we initialized α_0 to equal 0.5, a relatively large value. Then we defined $\alpha_k = \alpha_0 * 2^{-\eta_k}$, where η_k is the number of times that $L(u^{(k')}) > L(u^{(k'-1)})$ for $k' \leq k$. This learning rate drops at a rate of $1/2^t$, where t is the number of times that the dual increases from one iteration to the next. See Koo et al. (2010) for a similar, but less aggressive step size used to solve a different task.

Acknowledgments MIT gratefully acknowledges the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the DARPA, AFRL, or the US government. Alexander Rush was supported under the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-C-0022. David Sontag was supported by a Google PhD Fellowship.

References

Y. Bar-Hillel, M. Perles, and E. Shamir. 1964. On formal properties of simple phrase structure grammars. In *Language and Information: Selected Essays on their Theory and Application*, pages 116–150.

X. Carreras, M. Collins, and T. Koo. 2008. TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proc CONLL*, pages 9–16.

X. Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proc. CoNLL*, pages 957–961.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP*, page 8.

M. Collins. 2003. Head-driven statistical models for natural language parsing. In *Computational linguistics*, volume 29, pages 589–637.

G.B. Dantzig and P. Wolfe. 1960. Decomposition principle for linear programs. In *Operations research*, volume 8, pages 101–111.

J. Duchi, D. Tarlow, G. Elidan, and D. Koller. 2007. Using combinatorial optimization within max-product belief propagation. In *NIPS*, volume 19.

J. Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62.

A. Globerson and T. Jaakkola. 2007. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *NIPS*, volume 21.

N. Komodakis, N. Paragios, and G. Tziritas. 2007. MRF optimization via dual decomposition: Message-passing revisited. In *International Conference on Computer Vision*.

T. Koo, X. Carreras, and M. Collins. 2008. Simple semi-supervised dependency parsing. In *Proc. ACL/HLT*.

T. Koo, A.M. Rush, M. Collins, T. Jaakkola, and D. Sontag. 2010. Dual Decomposition for Parsing with Non-Projective Head Automata. In *Proc. EMNLP*, pages 63–70.

B.H. Korte and J. Vygen. 2008. *Combinatorial optimization: theory and algorithms*. Springer Verlag.

M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. In *Computational linguistics*, volume 19, pages 313–330.

R.K. Martin, R.L. Rardin, and B.A. Campbell. 1990. Polyhedral characterization of discrete dynamic programming. *Operations research*, 38(1):127–138.

A.F.T. Martins, N.A. Smith, and E.P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proc. ACL*.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. HLT/EMNLP*, pages 523–530.

Angelia Nedić and Asuman Ozdaglar. 2009. Approximate primal solutions and rate analysis for dual subgradient methods. *SIAM Journal on Optimization*, 19(4):1757–1780.

B. Pang and L. Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proc. ACL*.

S. Riedel and J. Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *Proc. EMNLP*, pages 129–137.

D. Roth and W. Yih. 2005. Integer linear programming inference for conditional random fields. In *Proc. ICML*, pages 737–744.

Hanif D. Sherali and Warren P. Adams. 1994. A hierarchy of relaxations and convex hull characterizations for mixed-integer zero–one programming problems. *Discrete Applied Mathematics*, 52(1):83 – 106.

D.A. Smith and J. Eisner. 2008. Dependency parsing by belief propagation. In *Proc. EMNLP*, pages 145–156.

D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. 2008. Tightening LP relaxations for MAP using message passing. In *Proc. UAI*.

B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. 2004. Max-margin parsing. In *Proc. EMNLP*, pages 1–8.

K. Toutanova and C.D. Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proc. EMNLP*, pages 63–70.

M. Wainwright and M. I. Jordan. 2008. *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers Inc., Hanover, MA, USA.

M. Wainwright, T. Jaakkola, and A. Willsky. 2005a. MAP estimation via agreement on trees: message-passing and linear programming. In *IEEE Transactions on Information Theory*, volume 51, pages 3697–3717.

M. Wainwright, T. Jaakkola, and A. Willsky. 2005b. A new class of upper bounds on the log partition function. In *IEEE Transactions on Information Theory*, volume 51, pages 2313–2335.

C. Yanover, T. Meltzer, and Y. Weiss. 2006. Linear Programming Relaxations and Belief Propagation–An Empirical Study. In *The Journal of Machine Learning Research*, volume 7, page 1907. MIT Press.

Self-training with Products of Latent Variable Grammars

Zhongqiang Huang[†]
[†]UMIACS
University of Maryland
College Park, MD
zqhuang@umd.edu

Mary Harper[†‡]
[‡]HLT Center of Excellence
Johns Hopkins University
Baltimore, MD
mharper@umd.edu

Slav Petrov[*]
[*]Google Research
76 Ninth Avenue
New York, NY
slav@google.com

Abstract

We study self-training with products of latent variable grammars in this paper. We show that increasing the quality of the automatically parsed data used for self-training gives higher accuracy self-trained grammars. Our generative self-trained grammars reach F scores of 91.6 on the WSJ test set and surpass even discriminative reranking systems without self-training. Additionally, we show that multiple self-trained grammars can be combined in a product model to achieve even higher accuracy. The product model is most effective when the individual underlying grammars are most diverse. Combining multiple grammars that were self-trained on disjoint sets of unlabeled data results in a final test accuracy of 92.5% on the WSJ test set and 89.6% on our Broadcast News test set.

1 Introduction

The latent variable approach of Petrov et al. (2006) is capable of learning high accuracy context-free grammars directly from a raw treebank. It starts from a coarse treebank grammar (Charniak, 1997), and uses latent variables to refine the context-free assumptions encoded in the grammar. A hierarchical split-and-merge algorithm introduces grammar complexity gradually, iteratively splitting (and potentially merging back) each observed treebank category into a number of increasingly refined latent subcategories. The Expectation Maximization (EM) algorithm is used to train the model, guaranteeing that each EM iteration will increase the training likelihood. However, because the latent variable grammars are not explicitly regularized, EM keeps fitting the training data and eventually begins overfitting (Liang et al., 2007). Moreover, EM is a local method, making no promises regarding the final point of convergence when initialized from different random seeds. Recently, Petrov (2010) showed that substantial differences between the learned grammars remain, even if the hierarchical splitting reduces the variance across independent runs of EM.

In order to counteract the overfitting behavior, Petrov et al. (2006) introduced a linear smoothing procedure that allows training grammars for 6 split-merge (SM) rounds without overfitting. The increased expressiveness of the model, combined with the more robust parameter estimates provided by the smoothing, results in a nice increase in parsing accuracy on a held-out set. However, as reported by Petrov (2009) and Huang and Harper (2009), an additional 7th SM round actually hurts performance.

Huang and Harper (2009) addressed the issue of data sparsity and overfitting from a different angle. They showed that self-training latent variable grammars on their own output can mitigate data sparsity issues and improve parsing accuracy. Because the capacity of the model can grow with the size of the training data, latent variable grammars are able to benefit from the additional training data, even though it is not perfectly labeled. Consequently, they also found that a 7th round of SM training was beneficial in the presence of large amounts of training data. However, variation still remains in their self-trained grammars and they had to use a held-out set for model selection.

The observation of variation is not surprising; EM's tendency to get stuck in local maxima has been studied extensively in the literature, resulting in various proposals for model selection methods (e.g., see

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 12–22,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

Burnham and Anderson (2002)). What is perhaps more surprising is that the different latent variable grammars seem to capture complementary aspects of the data. Petrov (2010) showed that a simple randomization scheme produces widely varying grammars. Quite serendipitously, these grammars can be combined into an unweighted product model that substantially outperforms the individual grammars.

In this paper, we combine the ideas of self-training and product models and show that both techniques provide complementary effects. We hypothesize that the main factors contributing to the final accuracy of the product model of self-trained grammars are (i) the accuracy of the grammar used to parse the unlabeled data for retraining (single grammar versus product of grammars) and (ii) the diversity of the grammars that are being combined (self-trained grammars trained using the same automatically labeled subset or different subsets). We conduct a series of analyses to develop an understanding of these factors, and conclude that both dimensions are important for obtaining significant improvements over the standard product models.

2 Experimental Setup

2.1 Data

We conducted experiments in two genres: newswire text and broadcast news transcripts. For the newswire studies, we used the standard setup (sections 02-21 for training, 22 for development, and 23 for final test) of the WSJ Penn Treebank (Marcus et al., 1999) for supervised training. The BLLIP corpus (Charniak et al., 2000) was used as a source of unlabeled data for self-training the WSJ grammars. We ignored the parse trees contained in the BLLIP corpus and retained only the sentences, which are already segmented and tokenized for parsing (e.g., contractions are split into two tokens and punctuation is separated from the words). We partitioned the 1,769,055 BLLIP sentences into 10 equally sized subsets[1].

For broadcast news (BN), we utilized the Broadcast News treebank from Ontonotes (Weischedel et al., 2008) together with the WSJ Penn Treebank for supervised training, because their combination results in better parser models compared to using the limited-sized BN corpus alone (86.7 F vs. 85.2 F). The files in the Broadcast News treebank represent news stories collected during different time periods with a diversity of topics. In order to obtain a representative split of train-test-development sets, we divided them into blocks of 10 files sorted by alphabetical filename order. We used the first file in each block for development, the second for test, and the remaining files for training. This training set was then combined with the entire WSJ treebank. We also used 10 equally sized subsets from the Hub4 CSR 1996 utterances (Garofolo et al., 1996) for self-training. The Hub 4 transcripts are markedly noisier than the BLLIP corpus is, in part because it is harder to sentence segment, but also because it was produced by human transcription of spoken language.

The treebanks were pre-processed differently for the two genres. For newswire, we used a slightly modified version of the WSJ treebank: empty nodes and function labels were deleted and auxiliary verbs were replaced with AUXB, AUXG, AUXZ, AUXD, or AUXN to represent infinitive, progressive, present, past, or past participle auxiliaries[2]. The targeted use of the broadcast models is for parsing broadcast news transcripts for language models in speech recognition systems. Therefore, in addition to applying the transformations used for newswire, we also replaced symbolic expressions with verbal forms (e.g., $5 was replaced with five dollars) and removed punctuation and case. The Hub4 data was segmented into utterances, punctuation was removed, words were down-cased, and contractions were tokenized for parsing. Table 1 summarizes the data set sizes used in our experiments, together with average sentence length and standard deviation.

2.2 Scoring

Parses from all models are compared with respective gold standard parses using SParseval bracket scoring (Roark et al., 2006). This scoring tool pro-

[1] We corrected some of the most egregious sentence segmentation problems in this corpus, and so the number of sentences is different than if one simply pulled the fringe of the trees. It was not uncommon for a sentence split to occur on abbreviations, such as Adm.

[2] Parsing accuracy is marginally affected. The average over 10 SM6 grammars with the transformation is 90.5 compared to 90.4 F without it, a 0.1% average improvement.

Genre	Statistics	Train	Dev	Test	Unlabeled
Newswire	# sentences	39.8k	1.7k	2.4k	1,769.1k
	# words	950.0k	40.1k	56.7k	43,057.0k
	length Avg./Std.	28.9/11.2	25.1/11.8	25.1/12.0	24.3/10.9
Broadcast News	# sentences	59.0k	1.0k	1.1k	4,386.5k
	# words	1,281.1k	17.1k	19.4k	77,687.9k
	length Avg./Std.	17.3/11.3	17.4/11.3	17.7/11.4	17.7/12.8

Table 1: The number of words and sentences, together with average (Avg.) sentence length and its standard deviation (Std.), for the data sets used in our experiments.

duces scores that are identical to those produced by EVALB for WSJ. For Broadcast News, SParseval applies Charniak and Johnson's (Charniak and Johnson, 2001) scoring method for EDITED nodes[3]. Using this method, BN scores were slightly (.05-.1) lower than if EDITED constituents were treated like any other, as in EVALB.

2.3 Latent Variable Grammars

We use the latent variable grammar (Matsuzaki et al., 2005; Petrov et al., 2006) implementation of Huang and Harper (2009) in this work. Latent variable grammars augment the observed parse trees in the treebank with a latent variable at each tree node. This effectively splits each observed category into a set of latent subcategories. An EM-algorithm is used to fit the model by maximizing the joint likelihood of parse trees and sentences. To allocate the grammar complexity only where needed, a simple split-and-merge procedure is applied. In every split-merge (SM) round, each latent variable is first split in two and the model is re-estimated. A likelihood criterion is used to merge back the least useful splits (50% merge rate for these experiments). This iterative refinement proceeds for 7 rounds, at which point parsing performance on a held-out set levels off and training becomes prohibitively slow.

Since EM is a local method, different initializations will result in different grammars. In fact, Petrov (2010) recently showed that this EM-algorithm is very unstable and converges to widely varying local maxima. These local maxima correspond to different high quality latent variable grammars that have captured different types of patterns in the data. Because the individual models' mistakes are independent to some extent, multiple grammars can be effectively combined into an unweighted product model of much higher accuracy. We build upon this line of work and investigate methods to exploit products of latent variable grammars in the context of self-training.

3 Self-training Methodology

Different types of parser self-training have been proposed in the literature over the years. All of them involve parsing a set of unlabeled sentences with a baseline parser and then estimating a new parser by combining this automatically parsed data with the original training data. McClosky et al. (2006) presented a very effective method for self-training a two-stage parsing system consisting of a first-stage generative lexicalized parser and a second-stage discriminative reranker. In their approach, a large amount of unlabeled text is parsed by the two-stage system and the parameters of the first-stage lexicalized parser are then re-estimated taking the counts from the automatically parsed data into consideration.

More recently Huang and Harper (2009) presented a self-training procedure based on an EM-algorithm. They showed that the EM-algorithm that is typically used to fit a latent variable grammar (Matsuzaki et al., 2005; Petrov et al., 2006) to a treebank can also be used for self-training on automatically parsed sentences. In this paper, we investigate self-training with products of latent variable grammars. We consider three training scenarios:

ST-Reg Training Use the best single grammar to

[3] Non-terminal subconstituents of EDITED nodes are removed so that the terminal constituents become immediate children of a single EDITED node, adjacent EDITED nodes are merged, and they are ignored for span calculations of the other constituents.

Regular	Best	Average	Product
SM6	90.8	90.5	92.0
SM7	90.4	90.1	92.2

Table 2: Performance of the regular grammars and their products on the WSJ development set.

ST-Reg	Best	Average	Product
SM6	91.5	91.2	92.0
SM7	91.6	91.5	92.4

Table 3: Performance of the ST-Reg grammars and their products on the WSJ development set.

parse a single subset of the unlabeled data and train 10 self-trained grammars using this single set.

ST-Prod Training Use the product model to parse a single subset of the unlabeled data and train 10 self-trained grammars using this single set.

ST-Prod-Mult Training Use the product model to parse all 10 subsets of the unlabeled data and train 10 self-trained grammars, each using a different subset.

The resulting grammars can be either used individually or combined in a product model.

These three conditions provide different insights. The first experiment allows us to investigate the effectiveness of product models for standard self-trained grammars. The second experiment enables us to quantify how important the accuracy of the baseline parser is for self-training. Finally, the third experiment investigates a method for injecting some additional diversity into the individual grammars to determine whether a product model is most successful when there is more variance among the individual models.

Our initial experiments and analysis will focus on the development set of WSJ. We will then follow up with an analysis of broadcast news (BN) to determine whether the findings generalize to a second, less structured type of data. It is important to construct grammars capable of parsing this type of data accurately and consistently in order to support structured language modeling (e.g., (Wang and Harper, 2002; Filimonov and Harper, 2009)).

4 Newswire Experiments

In this section, we compare single grammars and their products that are trained in the standard way with gold WSJ training data, as well as the three self-training scenarios discussed in Section 3. We report the F scores of both SM6 and SM7 grammars on the development set in order to evaluate the effect of model complexity on the performance of the self-trained and product models. Note that we use 6th round grammars to produce the automatic parse trees for the self-training experiments. Parsing with the product of the 7th round grammars is slow and requires a large amount of memory (32GB). Since we had limited access to such machines, it was infeasible for us to parse all of the unlabeled data with the SM7 product grammars.

4.1 Regular Training

We begin by training ten latent variable models initialized with different random seeds using the gold WSJ training set. Results are presented in Table 2. The best F score attained by the individual SM6 grammars on the development set is 90.8, with an average score of 90.5. The product of grammars achieves a significantly improved accuracy at 92.0[4]. Notice that the individual SM7 grammars perform worse on average (90.1 vs. 90.5) due to overfitting, but their product achieves higher accuracy than the product of the SM6 grammars (92.2 vs. 92.0). We will further investigate the causes for this effect in Section 5.

4.2 ST-Reg Training

Given the ten SM6 grammars from the previous subsection, we can investigate the three self-training methods. In the first regime (ST-Reg), we use the best single grammar (90.8 F) to parse a single subset of the BLLIP data. We then train ten grammars from different random seeds, using an equally weighted combination of the WSJ training set with this single set. These self-trained grammars are then combined into a product model. As reported in Table 3,

[4] We use Dan Bikel's randomized parsing evaluation comparator to determine the significance ($p < 0.05$) of the difference between two parsers' outputs.

ST-Prod	Best	Average	Product
SM6	91.7	91.4	92.2
SM7	91.9	91.7	92.4

Table 4: Performance of the ST-Prod grammars and their products on the WSJ development set.

ST-Prod-Mult	Best	Average	Product
SM6	91.7	91.4	92.5
SM7	91.8	91.7	92.8

Table 5: Performance of the ST-Prod-Mult grammars and their products on the WSJ development set.

thanks to the use of additional automatically labeled training data, the individual SM6 ST-Reg grammars perform significantly better than the individual SM6 grammars (91.2 vs. 90.5 on average), and the individual SM7 ST-Reg grammars perform even better, achieving a high F score of 91.5 on average.

The product of ST-Reg grammars achieves significantly better performance over the individual grammars, however, the improvement is much smaller than that obtained by the product of regular grammars. In fact, the product of ST-Reg grammars performs quite similarly to the product of regular grammars despite the higher average accuracy of the individual grammars. This may be caused by the fact that self-training on the same data tends to reduce the variation among the self-trained grammars. We will show in Section 5 that the diversity among the individual grammars is as important as average accuracy for the performance attained by the product model.

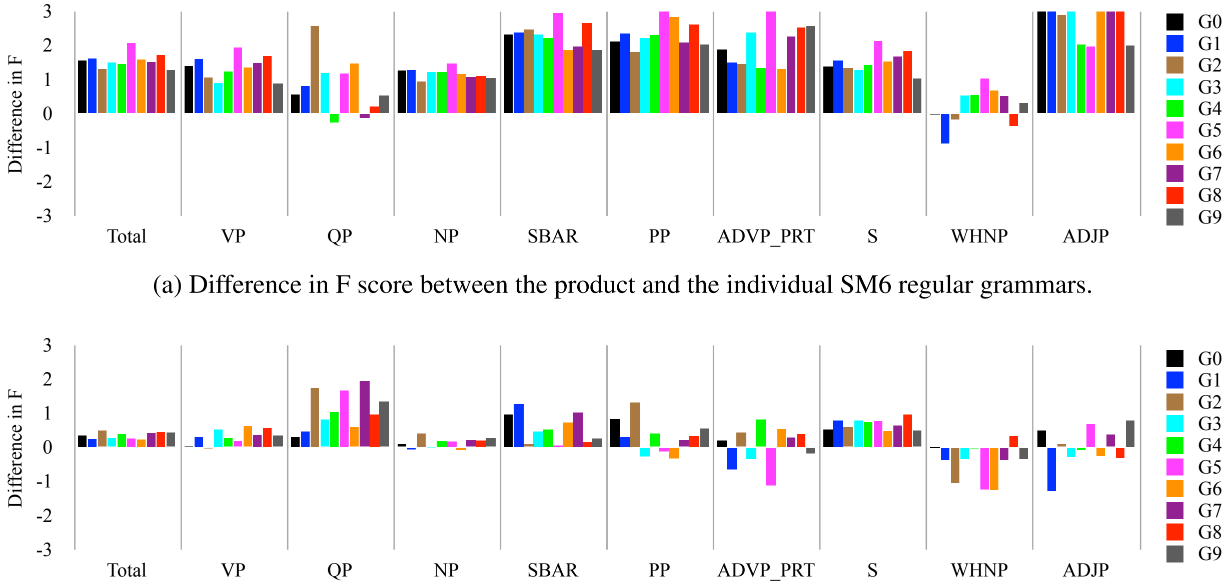
4.3 ST-Prod Training

Since products of latent variable grammars perform significantly better than individual latent variable grammars, it is natural to try using the product model for parsing the unlabeled data. To investigate whether the higher accuracy of the automatically labeled data translates into a higher accuracy of the self-trained grammars, we used the product of 6th round grammars to parse the same subset of the unlabeled data as in the previous experiment. We then trained ten self-trained grammars, which we call ST-Prod grammars. As can be seen in Table 4, using the product of the regular grammars for labeling the self-training data results in improved individual ST-Prod grammars when compared with the ST-Reg grammars, with 0.2 and 0.3 improvements for the best SM6 and SM7 grammars, respectively. Interestingly, the best individual SM7 ST-Prod grammar (91.9 F) performs comparably to the product of the regular grammars (92.0 F) that was used to label the BLLIP subset used for self-training. This is very useful for practical reasons because a single grammar is faster to parse with and requires less memory than the product model.

The product of the SM6 ST-Prod grammars also achieves a 0.2 higher F score compared to the product of the SM6 ST-Reg grammars, but the product of the SM7 ST-Prod grammars has the same performance as the product of the SM7 ST-Reg grammars. This could be due to the fact that the ST-Prod grammars are no more diverse than the ST-Reg grammars, as we will show in Section 5.

4.4 ST-Prod-Mult Training

When creating a product model of regular grammars, Petrov (2010) used a different random seed for each model and conjectured that the effectiveness of the product grammars stems from the resulting diversity of the individual grammars. Two ways to systematically introduce bias into individual models are to either modify the feature sets (Baldridge and Osborne, 2008; Smith and Osborne, 2007) or to change the training distributions of the individual models (Breiman, 1996). Petrov (2010) attempted to use the second method to train individual grammars on either disjoint or overlapping subsets of the treebank, but observed a performance drop in individual grammars resulting from training on less data, as well as in the performance of the product model. Rather than reducing the amount of gold training data (or having treebank experts annotate more data to support the diversity), we employ the self-training paradigm to train models using a combination of the same gold training data with different sets of the self-labeled training data. This approach also allows us to utilize a much larger amount of low-cost self-labeled data than can be used to train one model by partitioning the data into ten subsets and then training ten models with a different subset. Hence, in



(a) Difference in F score between the product and the individual SM6 regular grammars.

(b) Difference in F score between the product of SM6 regular grammars and the individual SM7 ST-Prod-Mult grammars.

Figure 1: Difference in F scores between various individual grammars and representative product grammars.

the third self-training experiment, we use the product of the regular grammars to parse all ten subsets of the unlabeled data and train ten grammars, which we call ST-Prod-Mult grammars, each using a different subset.

As shown in Table 5, the individual ST-Prod-Mult grammars perform similarly to the individual ST-Prod grammars. However, the product of the ST-Prod-Mult grammars achieves significantly higher accuracies than the product of the ST-Prod grammars, with 0.3 and 0.4 improvements for SM6 and SM7 grammars, respectively, suggesting that the use of multiple self-training subsets plays an important role in model combination.

5 Analysis

We conducted a series of analyses to develop an understanding of the factors affecting the effectiveness of combining self-training with product models.

5.1 What Has Improved?

Figure 1(a) depicts the difference between the product and the individual SM6 regular grammars on overall F score, as well as individual constituent F scores. As can be observed, there are significant variations among the individual grammars, and the product of the regular grammars improves almost all categories, with a few exceptions (some individual grammars do better on QP and WHNP constituents).

Figure 1(b) shows the difference between the product of the SM6 regular grammars and the individual SM7 ST-Prod-Mult grammars. Self-training dramatically improves the quality of single grammars. In most of the categories, some individual ST-Prod-Mult grammars perform comparably or slightly better than the product of SM6 regular grammars used to automatically label the unlabeled training set.

5.2 Overfitting vs. Smoothing

Figure 2(a) and 2(b) depict the learning curves of the regular and the ST-Prod-Mult grammars. As more latent variables are introduced through the iterative SM training algorithm, the modeling capacity of the grammars increases, leading to improved performance. However, the performance of the regular grammars drops after 6 SM rounds, as also previously observed in (Huang and Harper, 2009; Petrov, 2009), suggesting that the regular SM7 grammars have overfit the relatively small-sized gold training data. In contrast, the performance of the self-trained grammars continues to improve in the 7th SM round. Huang and Harper (2009) argued that the additional self-labeled training data adds a smoothing effect to the grammars, supporting an increase in model complexity without overfitting.

Although the performance of the individual grammars, both regular and self-trained, varies significantly and the product model consistently helps, there is a non-negligible difference between the improvement achieved by the two product models over their component grammars. The regular product model improves upon its individual grammars more than the ST-Prod-Mult product does in the later SM rounds, as illustrated by the relative error reduction curves in figures 2(a) and (b). In particular, the product of the SM7 regular grammars gains a remarkable 2.1% absolute improvement over the average performance of the individual regular SM7 grammars and 0.2% absolute over the product of the regular SM6 grammars, despite the fact that the individual regular SM7 grammars perform worse than the SM6 grammars. This suggests that the product model is able to effectively exploit less smooth, overfit grammars. We will examine this issue further in the next subsection.

5.3 Diversity

From the perspective of Products of Experts (Hinton, 1999) or Logarithmic Opinion Pools (Smith et al., 2005), each individual expert learns complementary aspects of the training data and the veto power of product models enforces that the joint prediction of their product has to be licensed by all individual experts. One possible explanation of the observation in the previous subsection is that with the addition of more latent variables, the individual grammars become more deeply specialized on certain aspects of the training data. This specialization leads to greater diversity in their prediction preferences, especially in the presence of a small training set. On the other hand, the self-labeled training set size is much larger, and so the specialization process is therefore slowed down.

Petrov (2010) showed that the individually learned grammars are indeed very diverse by looking at the distribution of latent annotations across the treebank categories, as well as the variation in overall and individual category F scores (see Figure 1). However, these measures do not directly relate to the diversity of the prediction preferences of the grammars, as we observed similar patterns in the regular and self-trained models.

Given a sentence s and a set of grammars $\mathcal{G} = \{G_1, \cdots, G_n\}$, recall that the decoding algorithm of the product model (Petrov, 2010) searches for the best tree T such that the following objective function is maximized:

$$\sum_{r \in T} \sum_{G \in \mathcal{G}} \log p(r|s, G)$$

where $\log p(r|s, G)$ is the log posterior probability of rule r given sentence s and grammar G. The power of the product model comes directly from the diversity in $\log p(r|s, G)$ among individual grammars. If there is little diversity, the individual grammars would make similar predictions and there would be little or no benefit from using a product model. We use the average empirical variance of the log posterior probabilities of the rules among the learned grammars over a held-out set S as a proxy of the diversity among the grammars:

$$\frac{\sum\limits_{s \in S} \sum\limits_{G \in \mathcal{G}} \sum\limits_{r \in \mathcal{R}(G,s)} p(r|s, G)\text{VAR}(\log(p(r|s, \mathcal{G})))}{\sum\limits_{s \in S} \sum\limits_{G \in \mathcal{G}} \sum\limits_{r \in \mathcal{R}(G,s)} p(r|s, G)}$$

where $\mathcal{R}(G, s)$ represents the set of rules extracted from the chart when parsing sentence s with grammar G, and $\text{VAR}(\log(p(r|s, \mathcal{G})))$ is the variance of $\log(p(r|s, G))$ among all grammars $G \in \mathcal{G}$.

Note that the average empirical variance is only an approximation of the diversity among grammars. In particular, this measure tends to be biased to produce larger numbers when the posterior probabilities of rules tend to be small, because small differences in probability produce large changes in the log scale. This happens for coarser grammars produced in early SM stages when there is more uncertainty about what rules to apply, with the rules remaining in the parsing chart having low probabilities overall.

As shown in Figure 2(c), the average variances all start at a high value and then drop, probably due to the aforementioned bias. However, as the SM iteration continues, the average variances increase despite the bias. More interestingly, the variance

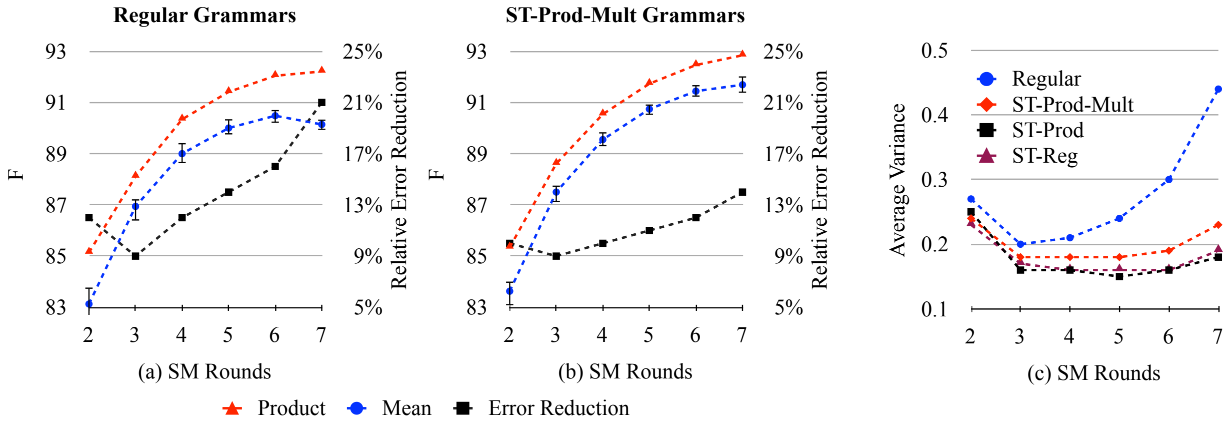


Figure 2: Learning curves of the individual regular (a) and ST-Prod-Mult (b) grammars (average performance, with minimum and maximum values indicated by bars) and their products before and after self-training on the WSJ development set. The relative error reductions of the products are also reported. (c) The measured average empirical variance among the grammars trained on WSJ.

among the regular grammars grows at a much faster speed and is consistently greater when compared to the self-trained grammars. This suggests that there is more diversity among the regular grammars than among the self-trained grammars, and explains the greater improvement obtained by the regular product model. It is also important to note that there is more variance among the ST-Prod-Mult grammars, which were trained on disjoint self-labeled training data, and a greater improvement in their product model relative to the ST-Reg and ST-Prod grammars, further supporting the diversity hypothesis. Last but not the least, the trend seems to indicate that the variance of the self-trained grammars would continue increasing if EM training was extended by a few more SM rounds, potentially resulting in even better product models. It is currently impractical to test this due to the dramatic increase in computational requirements for an SM8 product model, and so we leave it for future work.

5.4 Generalization to Broadcast News

We conducted the same set of experiments on the broadcast news data set. While the development set results in Table 6 show similar trends to the WSJ results, the benefits from the combination of self-training and product models appear even more pronounced here. The best single ST-Prod-Mult grammar (89.2 F) alone is able to outperform the product of SM7 regular grammars (88.9 F), and their product achieves another 0.7 absolute improvement, resulting in a significantly better accuracy at 89.9 F.

Model	Rounds	Best	Product
Regular	SM6	87.1	88.6
	SM7	87.1	88.9
ST-Prod	SM6	88.5	89.0
	SM7	89.0	89.6
ST-Prod-Mult	SM6	88.8	89.5
	SM7	**89.2**	**89.9**

Table 6: F-score for various models on the BN development set.

Figure 3 shows again that the benefits of self-training and product models are complementary and can be stacked. As can be observed, the self-trained grammars have increasing F scores as the split-merge rounds increase, while the regular grammars have a slight decrease in F score after round 6. In contrast to the newswire models, it appears that the individual ST-Prod-Mult grammars trained on broadcast news always perform comparably to the product of the regular grammars at all SM rounds, including the product of SM7 regular grammars. This is noteworthy, given that the ST-Prod-Mult grammars are trained on the output of the worse performing product of the SM6 regular grammars. One

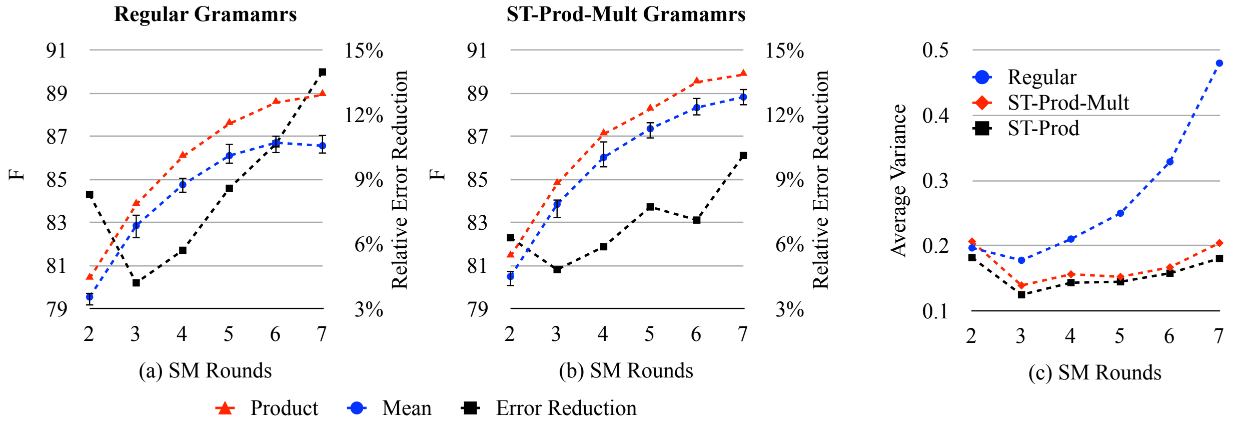


Figure 3: Learning curves of the individual regular (a) and ST-Prod-Mult (b) grammars (average performance, with minimum and maximum values indicated by bars) and their products before and after self-training on the BN development set. The relative error reductions of the products are also reported. (c) The measured average empirical variance among the grammars trained on BN.

possible explanation is that we used more unlabeled data for self-training the broadcast news grammars than for the newswire grammars. The product of the ST-Prod-Mult grammars provides further and significant improvement in F score.

6 Final Results

We evaluated the best single self-trained grammar (SM7 ST-Prod), as well as the product of the SM7 ST-Prod-Mult grammars on the WSJ test set. Table 7 compares these two grammars to a large body of related work grouped into single parsers (SINGLE), discriminative reranking approaches (RE), self-training (SELF), and system combinations (COMBO).

Our best single grammar achieves an accuracy that is only slightly worse (91.6 vs. 91.8 in F score) than the product model in Petrov (2010). This is made possible by self-training on the output of a high quality product model. The higher quality of the automatically parsed data results in a 0.3 point higher final F score (91.6 vs. 91.3) over the self-training results in Huang and Harper (2009), which used a single grammar for parsing the unlabeled data. The product of the self-trained ST-Prod-Mult grammars achieves significantly higher accuracies with an F score of 92.5, a 0.7 improvement over the product model in Petrov (2010).

Type	Parser	LP	LR	EX
SINGLE	Charniak (2000)	89.9	89.5	37.2
	Petrov and Klein (2007)	90.2	90.1	36.7
	Carreras et al. (2008)	91.4	90.7	-
RE	Charniak and Johnson (2005)	91.8	91.2	44.8
	Huang (2008)	92.2	91.2	43.5
SELF	Huang and Harper (2009)[8]	91.6	91.1	40.4
	McClosky et al. (2006)	92.5	92.1	45.3
COMBO	Petrov (2010)	92.0	91.7	41.9
	Sagae and Lavie (2006)	93.2	91.0	-
	Fossum and Knight (2009)	93.2	91.7	-
	Zhang et al. (2009)	93.3	92.0	-
	This Paper Best Single	91.8	91.4	40.3
	Best Product	92.7	92.2	43.1

Table 7: Final test set accuracies on WSJ.

Although our models are based on purely generative PCFG grammars, our best product model performs competitively to the self-trained two-step discriminative reranking parser of McClosky et al. (2006), which makes use of many non-local reranking features. Our parser also performs comparably to other system combination approaches (Sagae and Lavie, 2006; Fossum and Knight, 2009; Zhang et al., 2009) with higher recall and lower precision,

[8] Our ST-Reg grammars are trained in the same way as in Huang and Harper (2009) except that we keep all unary rules. The reported numbers are from the best single ST-Reg grammar in this work.

but again without using a discriminative reranking step. We expect that replacing the first-step generative parsing model in McClosky et al. (2006) with a product of latent variable grammars would give even higher parsing accuracies.

On the Broadcast News test set, our best performing single and product grammars (bolded in Table 6) obtained F scores of 88.7 and 89.6, respectively. While there is no prior work using our setup, we expect these numbers to set a high baseline.

7 Conclusions and Future Work

We evaluated methods for self-training high accuracy products of latent variable grammars with large amounts of genre-matched data. We demonstrated empirically on newswire and broadcast news genres that very high accuracies can be achieved by training grammars on disjoint sets of automatically labeled data. Two primary factors appear to be determining the efficacy of our self-training approach. First, the accuracy of the model used for parsing the unlabeled data is important for the accuracy of the resulting single self-trained grammars. Second, the diversity of the individual grammars controls the gains that can be obtained by combining multiple grammars into a product model. Our most accurate single grammar achieves an F score of 91.6 on the WSJ test set, rivaling discriminative reranking approaches (Charniak and Johnson, 2005) and products of latent variable grammars (Petrov, 2010), despite being a single generative PCFG. Our most accurate product model achieves an F score of 92.5 without the use of discriminative reranking and comes close to the best known numbers on this test set (Zhang et al., 2009).

In future work, we plan to investigate additional methods for increasing the diversity of our self-trained models. One possibility would be to utilize more unlabeled data or to identify additional ways to bias the models. It would also be interesting to determine whether further increasing the accuracy of the model used for automatically labeling the unlabeled data can enhance performance even more. A simple but computationally expensive way to do this would be to parse the data with an SM7 product model.

Finally, for this work, we always used products of 10 grammars, but we sometimes observed that subsets of these grammars produce even better results on the development set. Finding a way to select grammars from a grammar pool to achieve high performance products is an interesting area of future study.

8 Acknowledgments

This research was supported in part by NSF IIS-0703859. Opinions, findings, and recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agency or the institutions where the work was completed.

References

Jason Baldridge and Miles Osborne. 2008. Active learning and logarithmic opinion pools for HPSG parse selection. *Natural Language Engineering*.

Leo Breiman. 1996. Bagging predictors. *Machine Learning*.

Kenneth P. Burnham and David R. Anderson. 2002. *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. New York: Springer-Verlag.

Xavier Carreras, Michael Collins, and Terry Koo. 2008. Tag, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *CoNLL*, pages 9–16.

Eugene Charniak and Mark Johnson. 2001. Edit detection and parsing for transcribed speech. In *NAACL*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *ACL*.

Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, and Mark Johnson, 2000. *BLLIP 1987-89 WSJ Corpus Release 1*. Linguistic Data Consortium, Philadelphia.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *ICAI*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *ACL*.

Denis Filimonov and Mary Harper. 2009. A joint language model with fine-grain syntactic tags. In *EMNLP*, pages 1114–1123, Singapore, August.

Victoria Fossum and Kevin Knight. 2009. Combining constituent parsers. In *NAACL*, pages 253–256.

John Garofolo, Jonathan Fiscus, William Fisher, and David Pallett, 1996. *CSR-IV HUB4*. Linguistic Data Consortium, Philadelphia.

Geoffrey E. Hinton. 1999. Products of experts. In *ICANN*.

Zhongqiang Huang and Mary Harper. 2009. Self-training PCFG grammars with latent annotations across languages. In *EMNLP*.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *ACL*.

Percy Liang, Slav Petrov, Michael I. Jordan, and Dan Klein. 2007. The infinite PCFG using hierarchical Dirichlet processes. In *EMNLP*.

Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor, 1999. *Treebank-3*. Linguistic Data Consortium, Philadelphia.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *ACL*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *HLT-NAACL*.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *HLT-NAACL*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL*.

Slav Petrov. 2009. *Coarse-to-Fine Natural Language Processing*. Ph.D. thesis, University of California at Bekeley.

Slav Petrov. 2010. Products of random latent variable grammars. In *HLT-NAACL*.

Brian Roark, Mary Harper, Yang Liu, Robin Stewart, Matthew Lease, Matthew Snover, Izhak Shafran, Bonnie J. Dorr, John Hale, Anna Krasnyanskaya, and Lisa Yung. 2006. SParseval: Evaluation metrics for parsing speech. In *LREC*.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *NAACL*, pages 129–132.

Andrew Smith and Miles Osborne. 2007. Diversity in logarithmic opinion pools. *Lingvisticae Investigationes*.

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

Wen Wang and Mary P. Harper. 2002. The superarv language model: Investigating the effectiveness of tightly integrating multiple knowledge sources. In *EMNLP*, pages 238–247, Philadelphia, July.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, and Ann Houston, 2008. *OntoNotes Release 2.0*. Linguistic Data Consortium, Philadelphia.

Hui Zhang, Min Zhang, Chew Lim Tan, and Haizhou Li. 2009. K-best combination of syntactic parsers. In *EMNLP*, pages 1552–1560.

Utilizing Extra-sentential Context for Parsing

Jackie Chi Kit Cheung and Gerald Penn
Department of Computer Science
University of Toronto
Toronto, ON, M5S 3G4, Canada
{jcheung,gpenn}@cs.toronto.edu

Abstract

Syntactic consistency is the preference to reuse a syntactic construction shortly after its appearance in a discourse. We present an analysis of the WSJ portion of the Penn Treebank, and show that syntactic consistency is pervasive across productions with various left-hand side nonterminals. Then, we implement a reranking constituent parser that makes use of extra-sentential context in its feature set. Using a linear-chain conditional random field, we improve parsing accuracy over the generative baseline parser on the Penn Treebank WSJ corpus, rivalling a similar model that does not make use of context. We show that the context-aware and the context-ignorant rerankers perform well on different subsets of the evaluation data, suggesting a combined approach would provide further improvement. We also compare parses made by models, and suggest that context can be useful for parsing by capturing structural dependencies between sentences as opposed to lexically governed dependencies.

1 Introduction

Recent corpus linguistics work has produced evidence of syntactic consistency, the preference to reuse a syntactic construction shortly after its appearance in a discourse (Gries, 2005; Dubey et al., 2005; Reitter, 2008). In addition, experimental studies have confirmed the existence of syntactic priming, the psycholinguistic phenomenon of syntactic consistency[1]. Both types of studies, however, have limited the constructions that are examined to particular syntactic constructions and alternations. For instance, Bock (1986) and Gries (2005) examine specific constructions such as the passive voice, dative alternation and particle placement in phrasal verbs, and Dubey et al. (2005) deal with the internal structure of noun phrases. In this work, we extend these results and present an analysis of the distribution of all syntactic productions in the Penn Treebank WSJ corpus. We provide evidence that syntactic consistency is a widespread phenomenon across productions of various types of *LHS* nonterminals, including all of the commonly occurring ones.

Despite this growing evidence that the probability of syntactic constructions is not independent of the extra-sentential context, current high-performance statistical parsers (e.g. (Petrov and Klein, 2007; McClosky et al., 2006; Finkel et al., 2008)) rely solely on intra-sentential features, considering the particular grammatical constructions and lexical items within the sentence being parsed. We address this by implementing a reranking parser which takes advantage of features based on the context surrounding the sentence. The reranker outperforms the generative baseline parser, and rivals a similar model that does not make use of context. We show that the context-aware and the context-ignorant models perform well on different subsets of the evaluation data, suggesting a feature set that combines the two models would provide further improvement. Analysis of the rerankings made provides cases where contextual information has clearly improved parsing per-

[1] Whether or not corpus-based studies of consistency have any bearing on syntactic priming as a reality in the human mind is a subject of debate. See (Pickering and Branigan, 1999) and (Gries, 2005) for opposing viewpoints.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 23–33,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

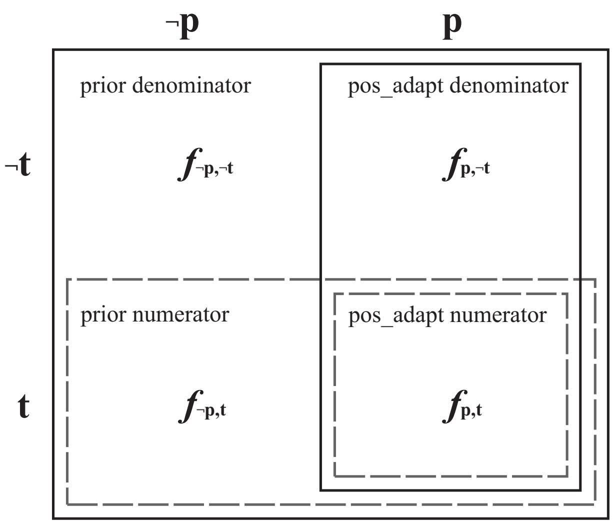


Figure 1: Visual representation of calculation of prior and positive adaptation probabilities. **t** represents the presence of a construction in the target set. **p** represents the presence of the construction in the prime set.

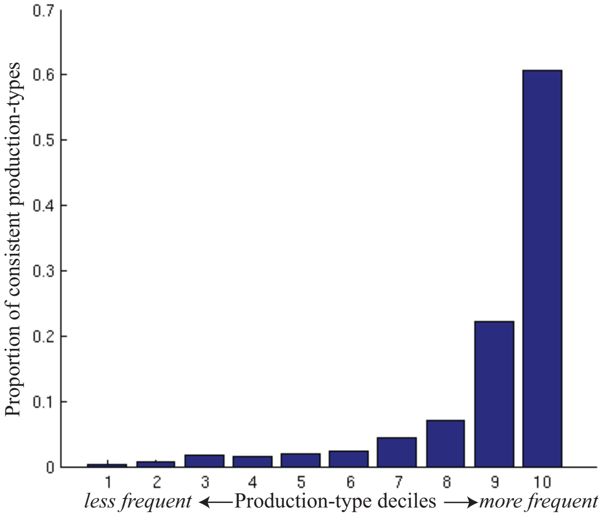


Figure 2: Production-types (singletons removed) categorized into deciles by frequency and the proportion of the production-types in that bin that is consistent to a significant degree.

formance, indicating the potential of extra-sentential contextual information to aid parsing, especially for structural dependencies between sentences, such as parallelism effects.

2 Syntactic Consistency in the Penn Treebank WSJ

Syntactic consistency has been examined by Dubey et al. (2005) for several English corpora, including the WSJ, Brown, and Switchboard corpora. They have provided evidence that syntactic consistency exists not only within coordinate structures, but also in a variety of other contexts, such as within sentences, between sentences, within documents, and between speaker turns in the Switchboard corpus. However, their analysis rests on a selected number of constructions concerning the internal structure of noun phrases. We extend their result here to arbitrary syntactic productions.

There have also been studies into syntactic consistency that consider all syntactic productions in dialogue corpora (Reitter, 2008; Buch and Pietsch, 2010). These studies find an inverse correlation between the probability of the appearance of a syntactic structure and the distance since its last occurrence, which indicates syntactic consistency. These studies, however, do not provide consistency results on subsets of production-types, such as by production LHS as our study does, so the implications that can be drawn from them for improving parsing are less apparent.

We adopt the measure used by Dubey et al. (2005) to quantify syntactic consistency, *adaptation probability*. This measure originates in work on lexical priming (Church, 2000), and quantifies the probability of a target word or construction w appearing in a "primed" context. Specifically, four frequencies are calculated, based on whether the target construction appears in the previous context (the prime set), and whether the construction appears after this context (the target set):

$$\begin{aligned} f_{p,\neg t}(w) &= \text{\# of times } w \text{ in prime set only} \\ f_{\neg p,t}(w) &= \text{\# of times } w \text{ in target set only} \\ f_{\neg p,\neg t}(w) &= \text{\# of times } w \text{ in neither set} \\ f_{p,t}(w) &= \text{\# of times } w \text{ in both sets} \end{aligned}$$

We also define N to be the sum of the four fre-

LHS	$prior$	pos_adapt	ratio	$+ > prior$ sig.	insig.	$+ < prior$ sig.
ADJP	0.03	0.05	1.96	26	251	0
ADVP	0.21	0.24	1.15	26	122	0
NP	0.17	0.22	1.27	281	2284	0
PP	0.56	0.58	1.04	32	125	0
PRN	0.01	0.03	4.60	12	82	0
PRT	0.06	0.08	1.40	3	3	0
QP	0.03	0.18	5.41	24	147	0
S	0.30	0.34	1.13	42	689	1
SBAR	0.15	0.20	1.31	13	68	0
SINV	0.01	0.01	1.00	3	77	0
VP	0.08	0.12	1.56	148	1459	0
WHADVP	0.04	0.08	1.84	2	8	0
WHNP	0.07	0.10	1.39	3	47	0
WHPP	0.01	0.02	2.65	1	1	0

Table 1: Weighted average by production frequency among non-singleton production-types of prior and positive adaptation probabilities, and the ratio between them. The columns on the right show the number of production-types for which the positive adaptation probability is significantly greater than, not different from, or less than the prior probability. We exclude *LHS*s with a weighted average prior of less than 0.005, due to the small sample size.

quencies. Then, we define the *prior* and the *positive adaptation* probability of a construction as follows (See also Figure 1):

$$\begin{aligned} prior(w) &= \frac{f_{p,t}(w) + f_{\neg p,t}(w)}{N} \\ pos\_adapt(w) &= \frac{f_{p,t}(w)}{f_{p,t}(w) + f_{p,\neg t}(w)} \end{aligned}$$

A positive adaptation probability that is greater than the prior probability would be interpreted as evidence for syntactic consistency for that construction. We conduct χ^2 tests for statistical significance testing. We analyze the Penn Treebank WSJ corpus according this schema for all productions that occur in sections 2 to 22. These are the standard training and development sets for training parsers. We did not analyze section 23 in order not to use its characteristics in designing our reranking parser so that we can use this section as our evaluation test set. Our analysis focuses on the consistency of rules between sentences, so we take the previous sentence within the same article as the prime set, and the current sentence as the target set in calculating the probabilities given above. The raw data from which we produced our analysis are available at `http://www.cs.toronto.edu/~jcheung/wsj_parallelism_data.txt`.

We first present results for consistency in all the production-types[2], grouped by the *LHS* of the production. Table 1 shows the weighted average prior and positive adaptation probabilities for productions by *LHS*, where the weighting is done by the number of occurrence of that production. Production-types that only occur once are removed. It also shows the number of production-types in which the positive adaptation probability is statistically significantly greater than, not significantly different from, and significantly lower than the prior probability.

Quite remarkably, very few production-types are significantly less likely to reoccur compared to the prior probability. Also note the wide variety of *LHS*s for which there is a large number of production-types that are consistent to a significant degree. While a large number of production-types appears not to be significantly more likely to occur in a primed context, this is due to the large number of production-types which only appear a few times. Frequently occurring production-types mostly exhibit syntactic consistency.

We show this in Figure 2, in which we put non-singleton production-types into ten bins by fre-

[2] That is, all occurrences of a production with a particular *LHS* and *RHS*.

Ten most frequent production-types						
production	$f_{\neg p,t}$	$f_{p,t}$	$f_{p,\neg t}$	$prior$	pos_adapt	ratio
PP → IN NP	5624	26224	5793	0.80	0.82	1.02
NP → NP PP	9033	12451	9388	0.54	0.57	1.05
NP → DT NN	9198	10585	9172	0.50	0.54	1.07
S → NP VP	8745	9897	9033	0.47	0.52	1.11
S → NP VP .	8576	8501	8888	0.43	0.49	1.13
S → VP	8717	7867	9042	0.42	0.47	1.11
NP → PRP	7208	5309	7285	0.32	0.42	1.33
ADVP → RB	7986	3949	7905	0.30	0.33	1.10
NP → NN	7630	3390	7568	0.28	0.31	1.11
VP → TO VP	7039	3552	7250	0.27	0.33	1.23

Ten most consistent among 10% most frequent production-types						
production	$f_{\neg p,t}$	$f_{p,t}$	$f_{p,\neg t}$	$prior$	pos_adapt	ratio
QP → # CD CD	51	18	45	0.00	0.29	163.85
NP → JJ NNPS	52	7	53	0.00	0.12	78.25
NP → NP , ADVP	109	24	99	0.00	0.20	58.05
NP → DT JJ CD NN	63	6	67	0.00	0.08	47.14
PP → IN NP NP	83	10	87	0.00	0.10	43.86
QP → IN $ CD	51	3	49	0.00	0.06	42.28
NP → NP : NP .	237	128	216	0.01	0.37	40.34
INTJ → UH	59	4	60	0.00	0.06	39.26
ADVP → IN NP	108	11	83	0.00	0.12	38.91
NP → CD CD	133	21	128	0.00	0.14	36.21

Table 2: Some instances of consistency effects of productions. All productions' pos_adapt probability is significantly greater than its $prior$ probability at $p < 10^{-6}$.

quency and calculated the proportion of production-types in that bin for which the positive adaptation probability is significantly greater than the prior. It is clear that the most frequently occurring production-types are also the ones most likely to exhibit evidence of syntactic consistency.

Table 2 shows the breakdown of the prior and positive adaptation calculation components for the ten most frequent production-types and the ten most consistent (by the ratio pos_adapt / $prior$) productions among the top decile of production-types. Note that all of these production-types are consistent to a statistically significant degree. Interestingly, many of the most consistent production-types have NP as the *LHS*, but overall, productions with many different *LHS* parents exhibit consistency.

3 A Context-Aware Reranker

Having established evidence for widespread syntactic consistency in the WSJ corpus, we now investigate incorporating extra-sentential context into a statistical parser. The first decision to make is whether to incorporate the context into a generative or a discriminative parsing model.

Employing a generative model would allow us to train the parser in one step, and one such parser which incorporates the previous context has been implemented by Dubey et al. (2006). They implement a PCFG, learning the production probabilities by a variant of standard PCFG-MLE probability estimation that conditions on whether a rule has recently occurred in the context or not:

$$P(RHS|LHS, Prime) = \frac{c(LHS \rightarrow RHS, Prime)}{c(LHS, Prime)}$$

LHS and *RHS* represent the left-hand side and right-hand side of a production, respectively. *Prime* is a binary variable which is `True` if and only if the current production has occurred in the prime set (the previous sentence). *c* represents the frequency count.

The drawback of such a system is that it doubles the state space of the model, and hence likely increases the amount of data needed to train the parser to a comparable level of performance as a more compact model, or would require elaborate smoothing. Dubey et al. (2006) find that this system performs worse than the baseline PCFG-MLE model, dropping F1 from 73.3% to 71.6%[3].

We instead opt to incorporate the extra-sentential context into a discriminative reranking parser, which naturally allows additional features to be incorporated into the statistical model. Many discriminative models of constituent parsing have been proposed in recent literature. They can be divided into two broad categories–those that rerank the N-best outputs of a generative parser, and those that make all parsing decisions using the discriminative model. We choose to implement an N-best reranking parser so that we can utilize state-of-the-art generative parsers to ensure a good selection of candidate parses to feed into our reranking module. Also, fully discriminative models tend to suffer from efficiency problems, though recent models have started to overcome this problem (Finkel et al., 2008).

Our approach is similar to N-best reranking parsers such as Charniak and Johnson (2005) and Collins and Koo (2005), which implement a variety of features to capture within-sentence lexical and structural dependencies. It is also similar to work which focuses on coordinate noun phrase parsing (e.g. (Hogan, 2007; Kübler et al., 2009)) in that we also attempt to exploit syntactic parallelism, but in a between-sentence setting rather than in a within-sentence setting that only considers coordination.

As evidence of the potential of an N-best reranking approach with respect to extra-sentential context, we considered the 50-best parses in the development set produced by the generative parser, and categorized each into one of nine bins depending on whether this candidate parse exhibits more, less, or the same amount of rule overlap with the previous correct parse than the generative baseline, and whether the candidate parse has a better, worse, or the same F1 measure than the generative baseline (Table 3). We find that a larger percentage of candidate parses which share more productions with the previous parse are better than the generative baseline parse than for the other categories, and this difference is statistically significant (χ^2 test).

	Overlap		
	less	*equal*	*more*
worse F1	32519 (81.8%)	7224 (69.3%)	17280 (75.4%)
equal F1	1023 (2.6%)	1674 (16.1%)	540 (2.4%)
better F1	6224 **(15.7%)**	1527 **(14.6%)**	5106 **(22.3%)**

Table 3: Correlation between rule overlap and F1 compared to the generative baseline for the 50-best parses in the development set.

3.1 Conditional Random Fields

For our statistical reranker, we implement a linear-chain conditional random field (CRF). CRFs are a very flexible class of graphical models which have been used for various sequence and relational labelling tasks (Lafferty et al., 2001). They have been used for tree labelling, in XML tree labelling (Jousse et al., 2006) and semantic role labelling tasks (Cohn and Blunsom, 2005). They have also been used for shallow parsing (Sha and Pereira, 2003), and full constituent parsing (Finkel et al., 2008; Tsuruoka et al., 2009). We exploit the flexibility of CRFs by incorporating features that depend on extra-sentential context.

In a linear-chain CRF, the conditional probability of a sequence of labels **y** = $y_{\{t=1...T\}}$ given a sequence of observed output **x** = $x_{\{t=1...T\}}$ and weight vector $\boldsymbol{\theta}$ = $\theta_{\{k=1...K\}}$ is given as follows:

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} exp(\sum_{t=1}^{T} \sum_{k} \theta_k f_k(y_{t-1}, y_t, \mathbf{x}, t))$$

[3] A similar model which conditions on whether productions have previously occurred *within the same sentence*, however, improves F1 to 73.6%.

where Z is the partition function. The feature functions $f_k(y_{t-1}, y_t, \mathbf{x}, t)$ can depend on two neighbouring parses, the sentences in the sequence, and the position of the sentence in the sequence. Since our feature functions do not depend on the words or the time-step within the sequence, however, we will write $f_k(y_{t-1}, y_t)$ from now on.

We treat each document in the corpus as one CRF sequence, and each sentence as one time-step in the sequence. The label sequence then is the sequence of parses, and the outputs are the sentences in the document. Since there is a large number of parses possible for each sentence and correspondingly many possible states for each label variable, we restrict the possible label state-space by extracting the N-best parses from a generative parser, and rerank over the sequences of candidate parses thus provided. We use the generative parser of Petrov and Klein (2007), a state-splitting parser that uses an EM algorithm to find splits in the nonterminal symbols to maximize training data likelihood. We use the 20-best parses, with an oracle F1 of 94.96% on section 23.

To learn the weight vector, we employ a stochastic gradient ascent method on the conditional log likelihood, which has been shown to perform well for parsing tasks (Finkel et al., 2008). In standard gradient ascent, the conditional log likelihood with a L2 regularization term for a Gaussian prior for a training corpus of N sequences is

$$\begin{aligned} L(\boldsymbol{\theta}) = & \sum_{i=1}^{N} \sum_{t,k} \theta_k f_k(y_{t-1}^{(i)}, y_t^{(i)}) \\ & - \sum_{i=1}^{N} log\ Z^{(i)} - \sum_{k} \frac{\theta_k^2}{2\sigma^2} \end{aligned}$$

And the partial derivatives with respect to the weights are

$$\begin{aligned} \frac{\partial L}{\partial \theta_k} = & \sum_{i=1}^{N} \sum_{t} f_k(y_{t-1}^{(i)}, y_t^{(i)}) \\ & - \sum_{i=1}^{N} \sum_{t} \sum_{y,y'} f_k(y, y') P(y, y' | \mathbf{x}^{(i)}) \\ & - \sum_{k} \frac{\theta_k}{\sigma^2} \end{aligned}$$

The first term is the feature counts in the training data, and the second term is the feature expectations according to the current weight vector. The third term corresponds to the penalty to non-zero weight values imposed by regularization. The probabilities in the second term can be efficiently calculated by the CRF-version of the forward-backward algorithm.

In standard gradient ascent, we update the weight vector after iterating through the whole training corpus. Because this is computationally expensive, we instead use stochastic gradient ascent, which approximates the true gradient by the gradient calculated from a single sample from the training corpus. We thus do not have to sum over the training set in the above expressions. We also employ a learning rate multiplier on the gradient. Thus, the weight update for the ith encountered training sequence during training is

$$\begin{aligned} \boldsymbol{\theta} &= \boldsymbol{\theta} + \alpha_i \nabla L_{stochastic}(\boldsymbol{\theta}) \\ \alpha_i &= \eta \frac{\tau \times N}{\tau \times N + i} \end{aligned}$$

The learning rate function is modelled on the one used by Finkel et al. (2008). It is designed such that α_i is halved after τ passes through the training set.

We train the model by iterating through the training set in a randomly permuted order, updating the weight vector after each sequence. The parameters η, τ, and σ are tuned to the development set. The final settings we use are $\eta = 0.08$, $\tau = 5$, and $\sigma = 50$. We use sections 2–21 of the Penn Treebank WSJ for training, 22 for development, and 23 for testing. We conduct 20-fold cross validation to generate the N-best parses for the training set, as is standard for N-best rerankers.

To rerank, we do inference with the linear-chain CRF for the most likely sequence of parses using the Viterbi algorithm.

3.2 Feature Functions

We experiment with various feature functions that depend on the syntactic and lexical parallelism between y_{t-1} and y_t. We use the occurrence of a rule in y_t that occurred in y_{t-1} as a feature. Based on the results of the corpus analysis, the first representation

(1) (S (NP (DT NN)) (VP (VBD)))

(2) (S (NP (NNS)) (VP (VBD)))

`Phrasal` features:
Template: (parent, child$_L$, child$_R$, repeated)
(S, edge, NP, $+$), (S, NP, VP, $+$), (S, VP, edge, +), (NP, edge, NNS, $-$), (NP, NNS, edge, $-$), (VP, edge, VBD, $+$), (VP, VBD, edge, $+$)

`Lexical` features:
Template: (parent, POS$_L$, POS$_R$, repeated)
(S, edge, NNS, $-$), (S, NNS, VBD, $-$), (S, VBD, edge, $+$), (NP, edge, NNS, $-$), (NP, NNS, edge, $-$), (VP, edge, VBD, $+$), (VP, VBD, edge, $+$)

Figure 3: Example of features extracted from a parse sequence specified down to the POS level.

Method	F1 (%)
Model-averaged	90.47
Combined, jointly trained $-$Context	90.33
Combined, jointly trained	90.31
Model-averaged $-$Context	90.22
`lexical` $-$Context	90.21
`lexical`	90.20
`phrasal`	90.12
`phrasal` $-$Context	89.74
Generative	89.70

Table 4: Development set (section 22) results of various models that we trained. Italicized are the models we use for the test set.

we tried was to simply enumerate the (non-lexical) productions in y_t along with whether that production is found in y_{t-1}. However, we found that our most successful feature function is to consider overlaps in partial structures of productions.

Specifically, we decompose a tree into all of the nonlexical vertically and horizontally markovized subtrees. Each of the subtrees in y_t marked by whether that same subtree occurs in the previous tree is a feature. The simple production representation corresponds to a vertical markovization of 1 and a horizontal markovization of infinite. We found that a vertical markovization of 1 and a horizontal markovization of 2 produced the best results on our data. We will call this model the `phrasal` model.

This schema so far only considers local substructures of parse trees, without being informed by the lexical information found in the leaves of the tree. We try another schema which considers the POS tag sequences found in each subtree. A feature then is the node label of the root of the subtree with the POS tag sequence it dominates, again decomposed into sequences of length 2 by markovization. We will call this model the `lexical` model.

To extract features from this sequence, we consider the substructures in the second parse, and mark whether they are found in the first parse as well. We add edge markers to mark the beginning and end of constituents. See Figure 3 for an example of features extracted by the two models.

We will consider various ways of combining the two schemata above in the next section. In addition, we also add a feature corresponding to the scaled log probability of a parse tree derived from the generative parsing baseline. Scaling is necessary because of the large differences in the magnitude of the log probability for different sentences. The scaling formula that we found to work best is to scale the maximum log probability among the N-best candidate parses to be 1.0 and the minimum to be 0.0.

3.3 Results

We train the two models which make use of extra-sentential context described in the previous section, and use the model to parse the development and test set. We also trained a model which combines both sets of features, but we found that we get better performance by training the two models separately, then averaging the models by computing the respective averages of their features' weights. Thus, we use the model-averaged version of the models that consider context in the test set experiments. The generative parser forms the first baseline method to which we compare our results. We also train a reranker which makes use of the same features as we described above, but without marking whether each substructure occurs in the previous sentence. This is thus a reranking method which does not make use of the previous context. Again, we tried model averaging, but this produces less accurate parses on the

	LP	LR	F1	Exact	$\overline{CB}$	0CB	LP	LR	F1	Exact	$\overline{CB}$	0CB
	development set – length $\leq$ 40						development set – all sentences					
Generative	90.33	90.20	90.27	39.92	0.68	71.99	89.64	89.75	89.70	37.76	0.82	68.65
$+$*Context*	**91.25**	90.71	**90.98**	**41.25**	**0.61**	**73.45**	**90.62**	90.33	**90.47**	**38.88**	**0.74**	**70.47**
$-$*Context*	90.85	**90.78**	90.82	40.62	0.62	73.00	90.28	**90.38**	90.22	38.24	**0.74**	70.00

Table 5: Parsing results on the development set (section 22) of the Penn Treebank WSJ (%, except for $\overline{CB}$). *Generative* is the generative baseline of Petrov and Klein (2007), $+$*Context* is the best performing reranking model using previous context (model-averaged `phrasal` and `lexical`), $-$*Context* is the best performing reranking model not using previous context (jointly trained `phrasal` and `lexical`).

	LP	LR	F1	Exact	$\overline{CB}$	0CB	LP	LR	F1	Exact	$\overline{CB}$	0CB
	test set – length $\leq$ 40						test set – all sentences					
Generative	90.04	89.84	89.94	38.31	0.80	68.33	89.60	89.35	89.47	36.05	0.94	65.81
$+$*Context*	90.63	90.11	90.37	**39.02**	0.73	69.40	90.17	89.64	89.91	**36.84**	0.87	67.09
$-$*Context*	**90.64**	**90.43**	**90.54**	38.62	**0.72**	**69.84**	**90.20**	**89.97**	**90.08**	36.47	**0.85**	**67.55**

Table 6: Parsing results on the test set (section 23) of the Penn Treebank WSJ (%, except for $\overline{CB}$)

development set, so we use the jointly trained model on the test set. We will refer to this model as the context-ignorant or $-$Context model, as opposed to the previous context-aware or $+$Context model. The results of these experiments on the development set are shown in Table 4.

PARSEVAL results[4] on the development and test set are presented in Tables 5 and 6. We see that the reranked models outperform the generative baseline model in terms of F1, and that the reranked model that uses extra-sentential context outperforms the version that does not use extra-sentential context in the development set, but not in the test set. Using Bikel's randomized parsing evaluation comparator[5], we find that both reranking models outperform the baseline generative model to statistical significance for recall and precision. The context-ignorant reranker outperforms the context-aware reranker on recall ($p < 0.01$), but not on precision ($p = 0.42$). However, the context-aware model has the highest exact match scores in both the development and the test set.

The F1 result suggests two possibilities–either the context-aware model captures the same information as the context-ignorant model, but less effectively, or the two models capture different information about the parses. Two pieces of evidence point to the latter possibility. First, if the context-aware model were truly inferior, then we would expect it to outperform the context-ignorant model on almost no sentences. Otherwise, we would expect them to do well on different sentences. Table 7 shows that the context-aware model outperforms the context-ignorant model on nearly as many trees in the test section as the reverse. Second, if we hypothetically had an oracle that could determine whether the context-ignorant or the context-aware model would be more accurate on a sentence and if the two models were complementary to each other, we would expect to achieve a gain in F1 over the generative baseline which is roughly the sum of the gain achieved by each model separately. This is indeed the case, as we are able to achieve F1s of 91.23% and 90.89% on sections 22 and 23 respectively, roughly twice the improvement that the individual models obtain.

Sec.	$-$*Context better*	*same*	$+$*Context better*
22	157	1357	186
23	258	1904	254

Table 7: Context-aware vs. context-ignorant reranking results, by sentential F1.

To put our results in perspective, we now compare the magnitude of the improvement in F1 our context-

[4] This evaluation ignores punctuation and corresponds to the `new.prm` parameter setting on evalb.

[5] `http://www.cis.upenn.edu/~dbikel/software.html`

System	*Baseline*	*Best*	*Imp. (rel.)*
Dubey et al. (2006)	73.3	73.6	0.3 (1.1%)
Hogan (2007)	89.4	89.6	0.2 (1.9%)
This work	**89.5**	**89.9**	**0.4 (3.8%)**

Table 8: A comparison of parsers specialized to exploit intra- or extra-sentential syntactic parallelism on section 23 in terms of the generative baseline they compare themselves against, the best F1 their non-baseline models achieve, and the absolute and relative improvements.

aware model achieves over the generative baseline to that of other systems specialized to exploit intra- or extra-sentential parallelism. We achieve a greater improvement despite the fact that our generative baseline provides a higher level of performance, and is presumably thus more difficult to improve upon (Table 8). These systems do not compare themselves against a reranked model that does not use parallelism as we do in this work.

During inference, the Viterbi algorithm recovers the most probable sequence of parses, and this means that we are relying on the generative parser to provide the context (i.e. the previous parses) when analyzing any given sentence. We do another type of oracle analysis in which we provide the parser with the correct, manually annotated parse tree of the previous sentence when extracting features for the current sentence during training and parsing. This "perfect context" model achieves F1s of 90.42% and 90.00% on sections 22 and 23 respectively, which is comparable to the best results of our reranking models. This indicates that the lack of perfect contextual information is not a major obstacle to further improving parsing performance.

3.4 Analysis

We now analyze several specific cases in the development set in which the reranker makes correct use of contextual information. They concretely illustrate how context can improve parsing performance, and confirm our initial intuition that extra-sentential context can be useful for parsing. The sentence in (3) and (4) is one such case.

(3) *Generative/Context-ignorant*: (S (S A BMA spokesman said "runaway medical costs" have made health insurance "a significant challenge) ," and (S margins also have been pinched ...) (. .))

(4) *Context-aware*: (S (NP A BMA spokesman) (VP said "runaway medical costs" have made health insurance "a significant challenge," and margins also have been pinched ...) (. .))

The baseline and the context-ignorant models parse the sentence as a conjunction of two S clauses, misanalyzing the scope of what is said by the BMA spokesman to the first part of the conjunct. By analyzing the features and feature weight values extracted from the parse sequence, we determined that the context-aware reranker is able to correct the analysis of the scoping due to a parallelism in the syntactic structure. Specifically, the substructure $S \rightarrow VP.$ is present in both this sentence and the previous sentence of the reranked sequence, which also contains a reporting verb.

(5) (S (NP BMA Corp., Kansas City, Mo.,) (VP said it's weighing "strategic alternatives" ... and is contacting possible buyers ...) (. .))

As a second example, consider the following sentence.

(6) *Generative/Context-ignorant*: To achieve maximum liquidity and minimize price volatility, (NP either all markets) (VP should be open to trading or none).

(7) *Context-aware*: To achieve maximum liquidity and minimize price volatility, (CC either) (S (NP all markets) should be open to trading or none).

The original generative and context-ignorant parses posit that "either all markets" is a noun phrase, which is incorrect. Syntactic parallelism corrects this for two reasons. First, the reranker prefers a determiner to start an NP in a consistent context, as both surround sentences also contain this substructure. Also, the previous sentence also contains a conjunction CC followed by a S node under a S node, which the reranker prefers.

While these examples show contextual features to be useful for parsing coordinations, we also found context-awareness to be useful for other types of structural ambiguity such as PP attachment ambiguity. Notice that the method we employ to correct coordination errors is different from previous approaches which usually rely on lexical or syntactic similarity between conjuncts rather than between sentences. Our approach can thus broaden the range of sentences that can be usefully reranked. For example, there is little similarity between conjuncts to avail of in the second example (Sentences 6 and 7).

Based on these analyses, it appears that context awareness provides a source of information for parsing which is not available to context-ignorant parsers. We should thus consider integrating both types of features into the reranking parser to build on the advantages of each. Specifically, within-sentence features are most appropriate for lexical dependencies and some structural dependencies. Extra-sentential features, on the other hand, are appropriate for capturing the syntactic consistency effects as we have demonstrated in this paper.

4 Conclusions

In this paper, we have examined evidence for syntactic consistency between neighbouring sentences. First, we conducted a corpus analysis of the Penn Treebank WSJ, and shown that parallelism exists between sentences for productions with a variety of *LHS* types, generalizing previous results for noun phrase structure. Then, we explored a novel source of features for parsing informed by the extra-sentential context. We improved on the parsing accuracy over a generative baseline parser, and rival a similar reranking model that does not rely on extra-sentential context. By examining the subsets of the evaluation data on which each model performs best and also individual cases, we argue that context allows a type of structural ambiguity resolution not available to parsers which only rely on intra-sentential context.

Acknowledgments

We would like to thank the anonymous reviewers and Timothy Fowler for their comments. This work is supported in part by the Natural Sciences and Engineering Research Council of Canada.

References

J.K. Bock. 1986. Syntactic persistence in language production. *Cognitive Psychology*, 18(3):355–387.

A. Buch and C. Pietsch. 2010. Measuring syntactic priming in dialog corpora. In *Proceedings of the Conference on Linguistic Evidence 2010: Empirical, Theoretical and Computational Perspectives*.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd ACL*, pages 173–180. Association for Computational Linguistics.

K.W. Church. 2000. Empirical estimates of adaptation: the chance of two Noriegas is closer to $p/2$ than p^2. In *Proceedings of 18th COLING*, pages 180–186. Association for Computational Linguistics.

T. Cohn and P. Blunsom. 2005. Semantic role labelling with tree conditional random fields. In *Ninth Conference on Computational Natural Language Learning*, pages 169–172.

M. Collins and T. Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70.

A. Dubey, P. Sturt, and F. Keller. 2005. Parallelism in coordination as an instance of syntactic priming: Evidence from corpus-based modeling. In *Proceedings of HLT/EMNLP 2005*, pages 827–834.

A. Dubey, F. Keller, and P. Sturt. 2006. Integrating syntactic priming into an incremental probabilistic parser, with an application to psycholinguistic modeling. In *Proceedings of the 21st COLING and the 44th ACL*, pages 417–424. Association for Computational Linguistics.

J.R. Finkel, A. Kleeman, and C.D. Manning. 2008. Efficient, feature-based, conditional random field parsing. *Proceedings of ACL-08: HLT*, pages 959–967.

S.T. Gries. 2005. Syntactic priming: A corpus-based approach. *Journal of Psycholinguistic Research*, 34(4):365–399.

D. Hogan. 2007. Coordinate noun phrase disambiguation in a generative parsing model. In *Proceedings of 45th ACL*, volume 45, pages 680–687.

F. Jousse, R. Gilleron, I. Tellier, and M. Tommasi. 2006. Conditional random fields for XML trees. In *ECML Workshop on Mining and Learning in Graphs*.

S. Kübler, W. Maier, E. Hinrichs, and E. Klett. 2009. Parsing coordinations. In *Proceedings of the 12th EACL*, pages 406–414. Association for Computational Linguistics.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, pages 282–289.

D. McClosky, E. Charniak, and M. Johnson. 2006. Effective self-training for parsing. In *Proceedings of HLT-NAACL 2006*.

S. Petrov and D. Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of HLT-NAACL 2007*, pages 404–411. Association for Computational Linguistics.

M.J. Pickering and H.P. Branigan. 1999. Syntactic priming in language production. *Trends in Cognitive Sciences*, 3(4):136–141.

D. Reitter. 2008. *Context Effects in Language Production: Models of Syntactic Priming in Dialogue Corpora*. Ph.D. thesis, University of Edinburgh.

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

Y. Tsuruoka, J. Tsujii, and S. Ananiadou. 2009. Fast full parsing by linear-chain conditional random fields. In *Proceedings of the 12th EACL*, pages 790–798. Association for Computational Linguistics.

Turbo Parsers: Dependency Parsing by Approximate Variational Inference

André F. T. Martins*† **Noah A. Smith*** **Eric P. Xing***
*School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{afm,nasmith,epxing}@cs.cmu.edu

Pedro M. Q. Aguiar‡
‡Instituto de Sistemas e Robótica
Instituto Superior Técnico
Lisboa, Portugal
aguiar@isr.ist.utl.pt

Mário A. T. Figueiredo†
†Instituto de Telecomunicações
Instituto Superior Técnico
Lisboa, Portugal
mtf@lx.it.pt

Abstract

We present a unified view of two state-of-the-art non-projective dependency parsers, both approximate: the loopy belief propagation parser of Smith and Eisner (2008) and the relaxed linear program of Martins et al. (2009). By representing the model assumptions with a factor graph, we shed light on the optimization problems tackled in each method. We also propose a new aggressive online algorithm to learn the model parameters, which makes use of the underlying variational representation. The algorithm does not require a learning rate parameter and provides a single framework for a wide family of convex loss functions, including CRFs and structured SVMs. Experiments show state-of-the-art performance for 14 languages.

1 Introduction

Feature-rich discriminative models that break locality/independence assumptions can boost a parser's performance (McDonald et al., 2006; Huang, 2008; Finkel et al., 2008; Smith and Eisner, 2008; Martins et al., 2009; Koo and Collins, 2010). Often, inference with such models becomes computationally intractable, causing a demand for understanding and improving approximate parsing algorithms.

In this paper, we show a formal connection between two recently-proposed approximate inference techniques for non-projective dependency parsing: loopy belief propagation (Smith and Eisner, 2008) and linear programming relaxation (Martins et al., 2009). While those two parsers are differently motivated, we show that both correspond to inference in a factor graph, and both optimize objective functions over local approximations of the marginal polytope. The connection is made clear by writing the explicit declarative optimization problem underlying Smith and Eisner (2008) and by showing the factor graph underlying Martins et al. (2009). The success of both approaches parallels similar approximations in other fields, such as statistical image processing and error-correcting coding. Throughtout, we call these *turbo parsers*.[1]

Our contributions are not limited to dependency parsing: we present a general method for inference in factor graphs with hard constraints (§2), which extends some combinatorial factors considered by Smith and Eisner (2008). After presenting a geometric view of the variational approximations underlying message-passing algorithms (§3), and closing the gap between the two aforementioned parsers (§4), we consider the problem of learning the model parameters (§5). To this end, we propose an aggressive online algorithm that generalizes MIRA (Crammer et al., 2006) to arbitrary loss functions. We adopt a family of losses subsuming CRFs (Lafferty et al., 2001) and structured SVMs (Taskar et al., 2003; Tsochantaridis et al., 2004). Finally, we present a technique for including features not attested in the training data, allowing for richer models without substantial runtime costs. Our experiments (§6) show state-of-the-art performance on dependency parsing benchmarks.

[1] The name stems from "turbo codes," a class of high-performance error-correcting codes introduced by Berrou et al. (1993) for which decoding algorithms are equivalent to running belief propagation in a graph with loops (McEliece et al., 1998).

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 34–44,
MIT, Massachusetts, USA, 9-11 October 2010.

2 Structured Inference and Factor Graphs

Denote by $\mathcal{X}$ a set of input objects from which we want to infer some hidden structure conveyed in an output set $\mathcal{Y}$. Each input $x \in \mathcal{X}$ (e.g., a sentence) is associated with a set of candidate outputs $\mathcal{Y}(x) \subseteq \mathcal{Y}$ (e.g., parse trees); we are interested in the case where $\mathcal{Y}(x)$ is a large structured set.

Choices about the *representation* of elements of $\mathcal{Y}(x)$ play a major role in algorithm design. In many problems, the elements of $\mathcal{Y}(x)$ can be represented as discrete-valued vectors of the form $\mathbf{y} = \langle y_1, \ldots, y_I \rangle$, each y_i taking values in a label set $\mathcal{Y}_i$. For example, in unlabeled dependency parsing, I is the number of candidate dependency arcs (quadratic in the sentence length), and each $\mathcal{Y}_i = \{0, 1\}$. Of course, the y_i are highly interdependent.

Factor Graphs. Probabilistic models like CRFs (Lafferty et al., 2001) assume a factorization of the conditional distribution of Y,

$$\Pr(Y = \mathbf{y} \mid X = x) \ \propto \ \textstyle\prod_{C \in \mathcal{C}} \Psi_C(x, \mathbf{y}_C), \quad (1)$$

where each $C \subseteq \{1, \ldots, I\}$ is a *factor*, $\mathcal{C}$ is the set of factors, each $\mathbf{y}_C \triangleq \langle y_i \rangle_{i \in C}$ denotes a partial output assignment, and each Ψ_C is a nonnegative *potential function* that depends on the output only via its restriction to C. A *factor graph* (Kschischang et al., 2001) is a convenient representation for the factorization in Eq. 1: it is a bipartite graph $\mathcal{G}_x$ comprised of variable nodes $\{1, \ldots, I\}$ and factor nodes $C \in \mathcal{C}$, with an edge connecting the ith variable node and a factor node C iff $i \in C$. Hence, the factor graph $\mathcal{G}_x$ makes explicit the direct dependencies among the variables $\{y_1, \ldots, y_I\}$.

Factor graphs have been used for several NLP tasks, such as dependency parsing, segmentation, and co-reference resolution (Sutton et al., 2007; Smith and Eisner, 2008; McCallum et al., 2009).

Hard and Soft Constraint Factors. It may be the case that valid outputs are a *proper* subset of $\mathcal{Y}_1 \times \cdots \times \mathcal{Y}_I$—for example, in dependency parsing, the entries of the output vector $\mathbf{y}$ must jointly define a spanning tree. This requires *hard constraint factors* that rule out forbidden partial assignments by mapping them to zero potential values. See Table 1 for an inventory of hard constraint factors used in this paper. Factors that are not of this special kind are called *soft factors*, and have strictly positive potentials. We thus have a partition $\mathcal{C} = \mathcal{C}_{\text{hard}} \cup \mathcal{C}_{\text{soft}}$.

We let the soft factor potentials take the form $\Psi_C(x, \mathbf{y}_C) \triangleq \exp(\boldsymbol{\theta}^\top \boldsymbol{\phi}_C(x, \mathbf{y}_C))$, where $\boldsymbol{\theta} \in \mathbb{R}^d$ is a vector of parameters (shared across factors) and $\boldsymbol{\phi}_C(x, \mathbf{y}_C)$ is a local feature vector. The conditional distribution of Y (Eq. 1) thus becomes log-linear:

$$\Pr_{\boldsymbol{\theta}}(\mathbf{y}|x) = Z_x(\boldsymbol{\theta})^{-1} \exp(\boldsymbol{\theta}^\top \boldsymbol{\phi}(x, \mathbf{y})), \quad (2)$$

where $Z_x(\boldsymbol{\theta}) \triangleq \sum_{\mathbf{y}' \in \mathcal{Y}(x)} \exp(\boldsymbol{\theta}^\top \boldsymbol{\phi}(x, \mathbf{y}'))$ is the *partition function*, and the features decompose as:

$$\boldsymbol{\phi}(x, \mathbf{y}) \triangleq \textstyle\sum_{C \in \mathcal{C}_{\text{soft}}} \boldsymbol{\phi}_C(x, \mathbf{y}_C). \quad (3)$$

Dependency Parsing. Smith and Eisner (2008) proposed a factor graph representation for dependency parsing (Fig. 1). The graph has $O(n^2)$ variable nodes (n is the sentence length), one per candidate arc $a \triangleq \langle h, m \rangle$ linking a head h and modifier m. Outputs are binary, with $y_a = 1$ iff arc a belongs to the dependency tree. There is a hard factor TREE connected to all variables, that constrains the overall arc configurations to form a spanning tree. There is a unary soft factor per arc, whose log-potential reflects the score of that arc. There are also $O(n^3)$ pairwise factors; their log-potentials reflect the scores of *sibling* and *grandparent* arcs. These factors create loops, thus calling for approximate inference. Without them, the model is arc-factored, and exact inference in it is well studied: finding the most probable parse tree takes $O(n^3)$ time with the Chu-Liu-Edmonds algorithm (McDonald et al., 2005),[2] and computing posterior marginals for all arcs takes $O(n^3)$ time via the matrix-tree theorem (Smith and Smith, 2007; Koo et al., 2007).

Message-passing algorithms. In general factor graphs, both inference problems—obtaining the most probable output (the MAP) $\text{argmax}_{\mathbf{y} \in \mathcal{Y}(x)} \Pr_{\boldsymbol{\theta}}(\mathbf{y}|x)$, and computing the marginals $\Pr_{\boldsymbol{\theta}}(Y_i = y_i|x)$—can be addressed with the belief propagation (BP) algorithm (Pearl, 1988), which iteratively passes messages between variables and factors reflecting their local "beliefs."

[2] There is a faster but more involved $O(n^2)$ algorithm due to Tarjan (1977).

A general binary factor: $\Psi_C(v_1, \dots, v_n) = \begin{cases} 1 & v_1, \dots, v_n \in \mathcal{S}_C \\ 0 & \text{otherwise,} \end{cases}$ where $\mathcal{S}_C \subseteq \{0,1\}^n$. • Message-induced distribution: $\boldsymbol{\omega} \triangleq \langle m_{j \to C} \rangle_{j=1,\dots,n}$ • Partition function: $Z_C(\boldsymbol{\omega}) \triangleq \sum_{\langle v_1,\dots,v_n\rangle \in \mathcal{S}_C} \prod_{i=1}^n m_{i\to C}^{v_i}$ • Marginals: $\text{MARG}_i(\boldsymbol{\omega}) \triangleq \Pr_{\boldsymbol{\omega}}\{V_i = 1 \mid \langle V_1, \dots, V_n\rangle \in \mathcal{S}_C\}$ • Max-marginals: $\text{MAX-MARG}_{i,b}(\boldsymbol{\omega}) \triangleq \max_{\mathbf{v}\in\mathcal{S}_C} \Pr_{\boldsymbol{\omega}}(\mathbf{v} \mid v_i = b)$ • Sum-prod.: $m_{C\to i} = m_{i\to C}^{-1} \cdot \text{MARG}_i(\boldsymbol{\omega})/(1 - \text{MARG}_i(\boldsymbol{\omega}))$ • Max-prod.: $m_{C\to i} = m_{i\to C}^{-1} \cdot \text{MAX-MARG}_{i,1}(\boldsymbol{\omega})/\text{MAX-MARG}_{i,0}(\boldsymbol{\omega})$ • Local agreem. constr.: $\mathbf{z} \in \text{conv}\, \mathcal{S}_C$, where $\mathbf{z} = \langle \tau_i(1) \rangle_{i=1}^n$ • Entropy: $H_C = \log Z_C(\boldsymbol{\omega}) - \sum_{i=1}^n \text{MARG}_i(\boldsymbol{\omega}) \log m_{i\to C}$
TREE $\Psi_{\text{TREE}}(\langle y_a \rangle_{a\in A}) = \begin{cases} 1 & \mathbf{y} \in \mathcal{Y}_{\text{tree}} \text{ (i.e., } \{a \in A \mid y_a = 1\} \text{ is a directed spanning tree)} \\ 0 & \text{otherwise,} \end{cases}$ where A is the set of candidate arcs. • Partition function $Z_{\text{tree}}(\boldsymbol{\omega})$ and marginals $\langle \text{MARG}_a(\boldsymbol{\omega}) \rangle_{a\in A}$ computed via the matrix-tree theorem, with $\boldsymbol{\omega} \triangleq \langle m_{a\to\text{TREE}} \rangle_{a\in A}$ • Sum-prod.: $m_{\text{TREE}\to a} = m_{a\to\text{TREE}}^{-1} \cdot \text{MARG}_a(\boldsymbol{\omega})/(1 - \text{MARG}_a(\boldsymbol{\omega}))$ • Max-prod.: $m_{\text{TREE}\to a} = m_{a\to\text{TREE}}^{-1} \cdot \text{MAX-MARG}_{a,1}(\boldsymbol{\omega})/\text{MAX-MARG}_{a,0}(\boldsymbol{\omega})$, where $\text{MAX-MARG}_{a,b}(\boldsymbol{\omega}) \triangleq \max_{\mathbf{y}\in\mathcal{Y}_{\text{tree}}} \Pr_{\boldsymbol{\omega}}(\mathbf{y} \mid y_a = b)$ • Local agreem. constr.: $\mathbf{z} \in \mathcal{Z}_{\text{tree}}$, where $\mathcal{Z}_{\text{tree}} \triangleq \text{conv}\, \mathcal{Y}_{\text{tree}}$ is the arborescence polytope • Entropy: $H_{\text{tree}} = \log Z_{\text{tree}}(\boldsymbol{\omega}) - \sum_{a\in A} \text{MARG}_a(\boldsymbol{\omega}) \log m_{a\to\text{TREE}}$
XOR (“one-hot”) $\Psi_{\text{XOR}}(v_1, \dots, v_n) = \begin{cases} 1 & \sum_{i=1}^n v_i = 1 \\ 0 & \text{otherwise.} \end{cases}$ • Sum-prod.: $m_{\text{XOR}\to i} = \left(\sum_{j\neq i} m_{j\to\text{XOR}}\right)^{-1}$ • Max-prod.: $m_{\text{XOR}\to i} = \left(\max_{j\neq i} m_{j\to\text{XOR}}\right)^{-1}$ • Local agreem. constr.: $\sum_i z_i = 1$, $z_i \in [0,1], \forall i$ • $H_{\text{XOR}} = -\sum_i (m_{i\to\text{XOR}}/\sum_j m_{j\to\text{XOR}}) \log(m_{i\to\text{XOR}}/\sum_j m_{j\to\text{XOR}})$
OR $\Psi_{\text{OR}}(v_1, \dots, v_n) = \begin{cases} 1 & \sum_{i=1}^n v_i \geq 1 \\ 0 & \text{otherwise.} \end{cases}$ • Sum-prod.: $m_{\text{OR}\to i} = \left(1 - \prod_{j\neq i}(1 + m_{j\to\text{OR}})^{-1}\right)^{-1}$ • Max-prod.: $m_{\text{OR}\to i} = \max\{1, \min_{j\neq i} m_{j\to\text{OR}}^{-1}\}$ • Local agreem. constr.: $\sum_i z_i \geq 1$, $z_i \in [0,1], \forall i$
OR-WITH-OUTPUT $\Psi_{\text{OR-OUT}}(v_1, \dots, v_n) = \begin{cases} 1 & v_n = \bigvee_{i=1}^{n-1} v_i \\ 0 & \text{otherwise.} \end{cases}$ • Sum-prod.: $m_{\text{OR-OUT}\to i} = \begin{cases} \left(1 - (1 - m_{n\to\text{OR-OUT}}^{-1}) \prod_{j\neq i,n}(1 + m_{j\to\text{OR-OUT}})^{-1}\right)^{-1} & i < n \\ \prod_{j\neq n}(1 + m_{j\to\text{OR-OUT}}) - 1 & i = n. \end{cases}$ • Max-prod.: $m_{\text{OR-OUT}\to i} = \begin{cases} \min\left\{m_{n\to\text{OR-OUT}} \prod_{j\neq i,n} \max\{1, m_{j\to\text{OR-OUT}}\}, \max\{1, \min_{j\neq i,n} m_{j\to\text{OR-OUT}}^{-1}\}\right\} & i < n \\ \prod_{j\neq n} \max\{1, m_{j\to\text{OR-OUT}}\} \min\{1, \max_{j\neq n} m_{j\to\text{OR-OUT}}\} & i = n. \end{cases}$

Table 1: Hard constraint factors, their potentials, messages, and entropies. The top row shows expressions for a general binary factor: each outgoing message is computed from incoming *marginals* (in the sum-product case), or *max-marginals* (in the max-product case); the entropy of the factor (see §3) is computed from these marginals and the partition function; the local agreement constraints (§4) involve the *convex hull* of the set $\mathcal{S}_C$ of allowed configurations (see footnote 5). The TREE, XOR, OR and OR-WITH-OUTPUT factors allow tractable computation of all these quantities (rows 2–5). Two of these factors (TREE and XOR) had been proposed by Smith and Eisner (2008); we provide further information (max-product messages, entropies, and local agreement constraints). Factors OR and OR-WITH-OUTPUT are novel to the best of our knowledge. This inventory covers many cases, since the above formulae can be extended to the case where some inputs are *negated*: just replace the corresponding messages by their reciprocal, v_i by $1 - v_i$, etc. This allows building factors NAND (an OR factor with negated inputs), IMPLY (a 2-input OR with the first input negated), and XOR-WITH-OUTPUT (an XOR factor with the last input negated).

In sum-product BP, the messages take the form:[3]

$$M_{i\to C}(y_i) \propto \prod_{D\neq C} M_{D\to i}(y_i) \tag{4}$$

$$M_{C\to i}(y_i) \propto \sum_{\mathbf{y}_C\sim y_i} \Psi_C(\mathbf{y}_C) \prod_{j\neq i} M_{j\to C}(y_j). \tag{5}$$

In max-product BP, the summation in Eq. 5 is replaced by a maximization. Upon convergence, variable and factor beliefs are computed as:

$$\tau_i(y_i) \propto \prod_C M_{C\to i}(y_i) \tag{6}$$

$$\tau_C(\mathbf{y}_C) \propto \Psi_C(\mathbf{y}_C) \prod_i M_{i\to C}(y_i). \tag{7}$$

BP is exact when the factor graph is a tree: in the sum-product case, the beliefs in Eqs. 6–7 correspond to the true marginals, and in the max-product case, maximizing each $\tau_i(y_i)$ yields the MAP output. In graphs with loops, BP is an approximate method, not guaranteed to converge, nicknamed *loopy* BP. We highlight a variational perspective of loopy BP in §3; for now we consider algorithmic issues. Note that computing the factor-to-variable messages for each factor C (Eq. 5) requires a summation/maximization over exponentially many configurations. Fortunately, for all the hard constraint factors in rows 3–5 of Table 1, this computation can be done in linear time (and polynomial for the TREE factor)—this extends results presented in Smith and Eisner (2008).[4]

[3] We employ the standard $\sim$ notation, where a summation/maximization indexed by $\mathbf{y}_C \sim y_i$ means that it is over all $\mathbf{y}_C$ with the i-th component held fixed and set to y_i.

[4] The insight behind these speed-ups is that messages on binary-valued potentials can be expressed as $M_{C\to i}(y_i) \propto$

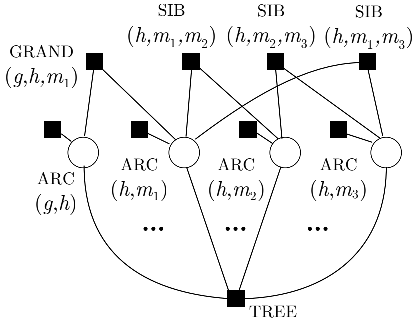


Figure 1: Factor graph corresponding to the dependency parsing model of Smith and Eisner (2008) with sibling and grandparent features. Circles denote variable nodes, and squares denote factor nodes. Note the loops created by the inclusion of pairwise factors (GRAND and SIB).

In Table 1 we present closed-form expressions for the factor-to-variable message ratios $m_{C\to i} \triangleq M_{C\to i}(1)/M_{C\to i}(0)$ in terms of their variable-to-factor counterparts $m_{i\to C} \triangleq M_{i\to C}(1)/M_{i\to C}(0)$; these ratios are all that is necessary when the variables are binary. Detailed derivations are presented in an extended version of this paper (Martins et al., 2010b).

3 Variational Representations

Let $\mathcal{P}_x \triangleq \{\Pr_{\boldsymbol{\theta}}(.|x) \mid \boldsymbol{\theta} \in \mathbb{R}^d\}$ be the family of all distributions of the form in Eq. 2. We next present an alternative parametrization for the distributions in $\mathcal{P}_x$ in terms of factor marginals. We will see that each distribution can be seen as a point in the so-called *marginal polytope* (Wainwright and Jordan, 2008); this will pave the way for the variational representations to be derived next.

Parts and Output Indicators. A *part* is a pair $\langle C, \mathbf{y}_C\rangle$, where C is a soft factor and $\mathbf{y}_C$ a partial output assignment. We let $\mathcal{R} = \{\langle C, \mathbf{y}_C\rangle \mid C \in \mathcal{C}_{\text{soft}}, \mathbf{y}_C \in \prod_{i\in C}\mathcal{Y}_i\}$ be the set of all parts. Given an output $\mathbf{y}' \in \mathcal{Y}(x)$, a part $\langle C, \mathbf{y}_C\rangle$ is said to be *active* if it locally matches the output, i.e., if $\mathbf{y}_C = \mathbf{y}'_C$. Any output $\mathbf{y}' \in \mathcal{Y}(x)$ can be mapped to a $|\mathcal{R}|$-dimensional binary vector $\boldsymbol{\chi}(\mathbf{y}')$ indicating which parts are active, i.e., $[\boldsymbol{\chi}(\mathbf{y}')]_{\langle C,\mathbf{y}_C\rangle} = 1$ if $\mathbf{y}_C = \mathbf{y}'_C$ and 0 otherwise; $\boldsymbol{\chi}(\mathbf{y}')$ is called the *output indicator vector*. This mapping allows decoupling the feature vector in Eq. 3 as the product of an input matrix and an output vector:

$$\boldsymbol{\phi}(x,\mathbf{y}) = \sum_{C\in\mathcal{C}_{\text{soft}}} \boldsymbol{\phi}_C(x,\mathbf{y}_C) = \mathbf{F}(x)\boldsymbol{\chi}(\mathbf{y}), \qquad (8)$$

where $\mathbf{F}(x)$ is a d-by-$|\mathcal{R}|$ matrix whose columns contain the part-local feature vectors $\boldsymbol{\phi}_C(x,\mathbf{y}_C)$. Observe, however, that not every vector in $\{0,1\}^{|\mathcal{R}|}$ corresponds necessarily to a valid output in $\mathcal{Y}(x)$.

Marginal Polytope. Moving to vector representations of outputs leads naturally to a geometric view of the problem. The *marginal polytope* is the convex hull[5] of all the "valid" output indicator vectors:

$$\mathcal{M}(\mathcal{G}_x) \triangleq \text{conv}\{\boldsymbol{\chi}(\mathbf{y}) \mid \mathbf{y} \in \mathcal{Y}(x)\}.$$

Note that $\mathcal{M}(\mathcal{G}_x)$ only depends on the factor graph $\mathcal{G}_x$ and the hard constraints (i.e., it is independent of the parameters $\boldsymbol{\theta}$). The importance of the marginal polytope stems from two facts: (i) each vertex of $\mathcal{M}(\mathcal{G}_x)$ corresponds to an output in $\mathcal{Y}(x)$; (ii) each point in $\mathcal{M}(\mathcal{G}_x)$ corresponds to a vector of marginal probabilities that is realizable by some distribution (not necessarily in $\mathcal{P}_x$) that factors according to $\mathcal{G}_x$.

Variational Representations. We now describe formally how the points in $\mathcal{M}(\mathcal{G}_x)$ are linked to the distributions in $\mathcal{P}_x$. We extend the "canonical overcomplete parametrization" case, studied by Wainwright and Jordan (2008), to our scenario (common in NLP), where arbitrary features are allowed and the parameters are *tied* (shared by all factors). Let $H(\Pr_{\boldsymbol{\theta}}(.|x)) \triangleq -\sum_{\mathbf{y}\in\mathcal{Y}(x)} \Pr_{\boldsymbol{\theta}}(\mathbf{y}|x)\log\Pr_{\boldsymbol{\theta}}(\mathbf{y}|x)$ denote the *entropy* of $\Pr_{\boldsymbol{\theta}}(.|x)$, and $E_{\boldsymbol{\theta}}[.]$ the expectation under $\Pr_{\boldsymbol{\theta}}(.|x)$. The component of $\boldsymbol{\mu} \in \mathcal{M}(\mathcal{G}_x)$ indexed by part $\langle C, \mathbf{y}_C\rangle$ is denoted $\mu_C(\mathbf{y}_C)$.

Proposition 1. *There is a map coupling each distribution* $\Pr_{\boldsymbol{\theta}}(.|x) \in \mathcal{P}_x$ *to a unique* $\boldsymbol{\mu} \in \mathcal{M}(\mathcal{G}_x)$ *such that* $E_{\boldsymbol{\theta}}[\boldsymbol{\chi}(Y)] = \boldsymbol{\mu}$. *Define* $H(\boldsymbol{\mu}) \triangleq H(\Pr_{\boldsymbol{\theta}}(.|x))$ *if some* $\Pr_{\boldsymbol{\theta}}(.|x)$ *is coupled to* $\boldsymbol{\mu}$, *and* $H(\boldsymbol{\mu}) = -\infty$ *if no such* $\Pr_{\boldsymbol{\theta}}(.|x)$ *exists. Then:*

1. *The following variational representation for the log-partition function (mentioned in Eq. 2) holds:*

$$\log Z_x(\boldsymbol{\theta}) = \max_{\boldsymbol{\mu}\in\mathcal{M}(\mathcal{G}_x)} \boldsymbol{\theta}^\top \mathbf{F}(x)\boldsymbol{\mu} + H(\boldsymbol{\mu}). \qquad (9)$$

$\Pr\{\Psi_C(Y_C) = 1|Y_i = y_i\}$ and $M_{C\to i}(y_i) \propto \max_{\Psi_C(\mathbf{y}_C)=1}\Pr\{Y_C = \mathbf{y}_C|Y_i = y_i\}$, respectively for the sum-product and max-product cases; these probabilities are induced by the messages in Eq. 4: for an event $A \subseteq \prod_{i\in C}\mathcal{Y}_i$, $\Pr\{Y_C \in A\} \triangleq \sum_{\mathbf{y}_C}\mathbb{I}(\mathbf{y}_C \in A)\prod_{i\in C} M_{i\to C}(y_i)$.

[5] The convex hull of $\{\mathbf{z}_1, \ldots, \mathbf{z}_k\}$ is the set of points that can be written as $\sum_{i=1}^k \lambda_i \mathbf{z}_i$, where $\sum_{i=1}^k \lambda_i = 1$ and each $\lambda_i \geq 0$.

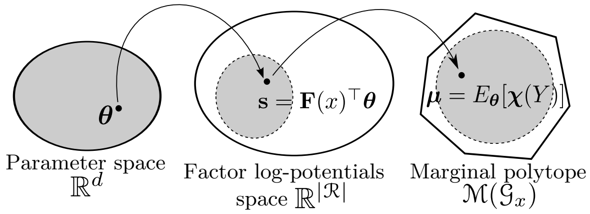

$\boldsymbol{\theta}$
$\mathbf{s} = \mathbf{F}(x)^\top\boldsymbol{\theta}$
$\boldsymbol{\mu} = E_{\boldsymbol{\theta}}[\boldsymbol{\chi}(Y)]$
Parameter space $\mathbb{R}^d$
Factor log-potentials space $\mathbb{R}^{|\mathcal{R}|}$
Marginal polytope $\mathcal{M}(\mathcal{G}_x)$

Figure 2: Dual parametrization of the distributions in $\mathcal{P}_x$. Our parameter space (left) is first linearly mapped to the space of factor log-potentials (middle). The latter is mapped to the marginal polytope $\mathcal{M}(\mathcal{G}_x)$ (right). In general only a subset of $\mathcal{M}(\mathcal{G}_x)$ is reachable from our parameter space. Any distribution in $\mathcal{P}_x$ can be parametrized by a vector $\boldsymbol{\theta} \in \mathbb{R}^d$ or by a point $\boldsymbol{\mu} \in \mathcal{M}(\mathcal{G}_x)$.

2. *The problem in Eq. 9 is convex and its solution is attained at the factor marginals, i.e., there is a maximizer $\bar{\boldsymbol{\mu}}$ s.t. $\bar{\mu}_C(\mathbf{y}_C) = \Pr_{\boldsymbol{\theta}}(Y_C = \mathbf{y}_C|x)$ for each $C \in \mathcal{C}$. The gradient of the log-partition function is $\nabla \log Z_x(\boldsymbol{\theta}) = \mathbf{F}(x)\bar{\boldsymbol{\mu}}$.*

3. *The MAP $\hat{\mathbf{y}} \triangleq \operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}(x)} \Pr_{\boldsymbol{\theta}}(\mathbf{y}|x)$ can be obtained by solving the linear program*

$$\hat{\boldsymbol{\mu}} \triangleq \boldsymbol{\chi}(\hat{\mathbf{y}}) = \operatorname*{argmax}_{\boldsymbol{\mu}\in\mathcal{M}(\mathcal{G}_x)} \boldsymbol{\theta}^\top \mathbf{F}(x)\boldsymbol{\mu}. \tag{10}$$

A proof of this proposition can be found in Martins et al. (2010a). Fig. 2 provides an illustration of the dual parametrization implied by Prop. 1.

4 Approximate Inference & Turbo Parsing

We now show how the variational machinery just described relates to message-passing algorithms and provides a common framework for analyzing two recent dependency parsers. Later (§5), Prop. 1 is used constructively for learning the model parameters.

4.1 Loopy BP as a Variational Approximation

For general factor graphs with loops, the marginal polytope $\mathcal{M}(\mathcal{G}_x)$ cannot be compactly specified and the entropy term $H(\boldsymbol{\mu})$ lacks a closed form, rendering exact optimizations in Eqs. 9–10 intractable. A popular *approximate* algorithm for marginal inference is sum-product loopy BP, which passes messages as described in §2 and, upon convergence, computes beliefs via Eqs. 6–7. Were loopy BP exact, these beliefs would be the true marginals and hence a point in the marginal polytope $\mathcal{M}(\mathcal{G}_x)$. However, this need not be the case, as elucidated by Yedidia et al. (2001) and others, who first analyzed loopy BP from a variational perspective. The following two approximations underlie loopy BP:

- The marginal polytope $\mathcal{M}(\mathcal{G}_x)$ is approximated by the *local polytope* $\mathcal{L}(\mathcal{G}_x)$. This is an *outer* bound; its name derives from the fact that it only imposes *local agreement constraints* $\forall i, y_i \in \mathcal{Y}_i, C \in \mathcal{C}$:

$$\textstyle\sum_{y_i} \tau_i(y_i) = 1, \ \sum_{\mathbf{y}_C \sim y_i} \tau_C(\mathbf{y}_C) = \tau_i(y_i). \tag{11}$$

  Namely, it is characterized by $\mathcal{L}(\mathcal{G}_x) \triangleq \{\boldsymbol{\tau} \in \mathbb{R}_+^{|\mathcal{R}|} \mid \text{Eq. 11 holds } \forall i, y_i \in \mathcal{Y}_i, C \in \mathcal{C}\}$. The elements of $\mathcal{L}(\mathcal{G}_x)$ are called *pseudo-marginals*. Clearly, the true marginals satisfy Eq. 11, and therefore $\mathcal{M}(\mathcal{G}_x) \subseteq \mathcal{L}(\mathcal{G}_x)$.

- The entropy H is replaced by its *Bethe approximation* $H_{\text{Bethe}}(\boldsymbol{\tau}) \triangleq \sum_{i=1}^{I}(1 - d_i)H(\boldsymbol{\tau}_i) + \sum_{C\in\mathcal{C}} H(\boldsymbol{\tau}_C)$, where $d_i = |\{C \mid i \in C\}|$ is the number of factors connected to the ith variable, $H(\boldsymbol{\tau}_i) \triangleq -\sum_{y_i} \tau_i(y_i) \log \tau_i(y_i)$ and $H(\boldsymbol{\tau}_C) \triangleq -\sum_{\mathbf{y}_C} \tau_C(\mathbf{y}_C) \log \tau_C(\mathbf{y}_C)$.

Any stationary point of sum-product BP is a local optimum of the variational problem in Eq. 9 with $\mathcal{M}(\mathcal{G}_x)$ replaced by $\mathcal{L}(\mathcal{G}_x)$ and H replaced by H_{Bethe} (Yedidia et al., 2001). Note however that multiple optima may exist, since H_{Bethe} is not necessarily concave, and that BP may not converge.

Table 1 shows closed form expressions for the local agreement constraints and entropies of some hard-constraint factors, obtained by invoking Eq. 7 and observing that $\tau_C(\mathbf{y}_C)$ must be zero if configuration $\mathbf{y}_C$ is forbidden. See Martins et al. (2010b).

4.2 Two Dependency Turbo Parsers

We next present our main contribution: a formal connection between two recent approximate dependency parsers, which at first sight appear unrelated. Recall that (i) Smith and Eisner (2008) proposed a factor graph (Fig. 1) in which they run loopy BP, and that (ii) Martins et al. (2009) approximate parsing as the solution of a linear program. Here, we fill the blanks in the two approaches: we derive explicitly the variational problem addressed in (i) and we provide the underlying factor graph in (ii). This puts the two approaches side-by-side as approximate methods for marginal and MAP inference. Since both rely on "local" approximations (in the sense of Eq. 11) that ignore the loops in their graphical models, we dub them *turbo parsers* by analogy with error-correcting turbo decoders (see footnote 1).

Turbo Parser #1: Sum-Product Loopy BP. The factor graph depicted in Fig. 1—call it $\mathcal{G}_x$—includes pairwise soft factors connecting sibling and grandparent arcs.[6] We next characterize the local polytope $\mathcal{L}(\mathcal{G}_x)$ and the Bethe approximation H_{Bethe} inherent in Smith and Eisner's loopy BP algorithm.

Let A be the set of candidate arcs, and $P \subseteq A^2$ the set of pairs of arcs that have factors. Let $\boldsymbol{\tau} = \langle \boldsymbol{\tau}_A, \boldsymbol{\tau}_P \rangle$ with $\boldsymbol{\tau}_A = \langle \tau_a \rangle_{a \in A}$ and $\boldsymbol{\tau}_P = \langle \tau_{ab} \rangle_{\langle a,b \rangle \in P}$. Since all variables are binary, we may write, for each $a \in A$, $\tau_a(1) = z_a$ and $\tau_a(0) = 1 - z_a$, where z_a is a variable constrained to $[0,1]$. Let $\mathbf{z}_A \triangleq \langle z_a \rangle_{a \in A}$; the local agreement constraints at the TREE factor (see Table 1) are written as $\mathbf{z}_A \in \mathcal{Z}_{\text{tree}}(x)$, where $\mathcal{Z}_{\text{tree}}(x)$ is the *arborescence polytope*, i.e., the convex hull of all incidence vectors of dependency trees (Martins et al., 2009). It is straightforward to write a contingency table and obtain the following local agreement constraints at the pairwise factors:

$$\begin{aligned} \tau_{ab}(1,1) &= z_{ab}, & \tau_{ab}(0,0) &= 1 - z_a - z_b + z_{ab} \\ \tau_{ab}(1,0) &= z_a - z_{ab}, & \tau_{ab}(0,1) &= z_b - z_{ab}. \end{aligned}$$

Noting that all these pseudo-marginals are constrained to the unit interval, one can get rid of all variables τ_{ab} and write everything as

$$\begin{aligned} &z_a \in [0,1], \quad z_b \in [0,1], \quad z_{ab} \in [0,1], \\ &z_{ab} \leq z_a, \quad\;\; z_{ab} \leq z_b, \quad\;\; z_{ab} \geq z_a + z_b - 1, \end{aligned} \tag{12}$$

inequalities which, along with $\mathbf{z}_A \in \mathcal{Z}_{\text{tree}}(x)$, define the local polytope $\mathcal{L}(\mathcal{G}_x)$. As for the factor entropies, start by noting that the TREE-factor entropy H_{tree} can be obtained in closed form by computing the marginals $\bar{\mathbf{z}}_A$ and the partition function $Z_x(\boldsymbol{\theta})$ (via the matrix-tree theorem) and recalling the variational representation in Eq. 9, yielding $H_{\text{tree}} = \log Z_x(\boldsymbol{\theta}) - \boldsymbol{\theta}^\top \mathbf{F}(x)\bar{\mathbf{z}}_A$. Some algebra allows writing the overall Bethe entropy approximation as:

$$H_{\text{Bethe}}(\boldsymbol{\tau}) = H_{\text{tree}}(\mathbf{z}_A) - \sum_{\langle a,b \rangle \in P} I_{a;b}(z_a, z_b, z_{ab}), \tag{13}$$

where we introduced the mutual information associated with each pairwise factor, $I_{a;b}(z_a, z_b, z_{ab}) = \sum_{y_a, y_b} \tau_{ab}(y_a, y_b) \log \frac{\tau_{ab}(y_a,y_b)}{\tau_a(y_a)\tau_b(y_b)}$. The approximate variational expression becomes $\log Z_x(\boldsymbol{\theta}) \approx$

$$\begin{aligned} \max_{\mathbf{z}} \quad & \boldsymbol{\theta}^\top \mathbf{F}(x)\mathbf{z} + H_{\text{tree}}(\mathbf{z}_A) - \sum_{\langle a,b \rangle \in P} I_{a;b}(z_a, z_b, z_{ab}) \\ \text{s.t.} \quad & z_{ab} \leq z_a, \quad z_{ab} \leq z_b, \\ & z_{ab} \geq z_a + z_b - 1, \quad \forall \langle a,b \rangle \in P, \\ & \mathbf{z}_A \in \mathcal{Z}_{\text{tree}}, \end{aligned} \tag{14}$$

whose maximizer corresponds to the beliefs returned by the Smith and Eisner's loopy BP algorithm (if it converges).

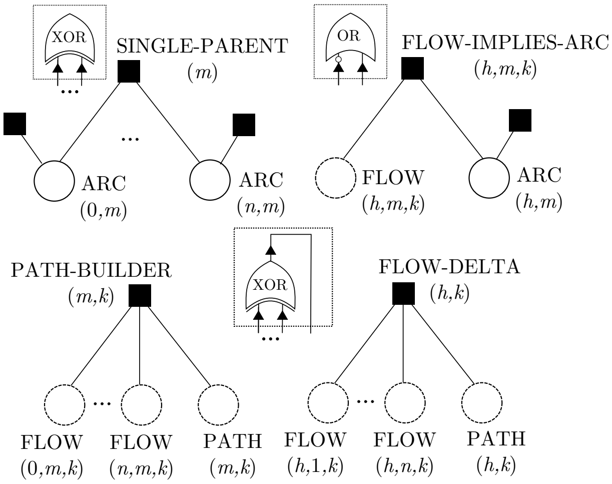


Figure 3: Details of the factor graph underlying the parser of Martins et al. (2009). Dashed circles represent auxiliary variables. See text and Table 1.

Turbo Parser #2: LP-Relaxed MAP. We now turn to the concise integer LP formulation of Martins et al. (2009). The formulation is exact but NP-hard, and so an LP relaxation is made there by dropping the integer constraints. We next construct a factor graph $\mathcal{G}'_x$ and show that the LP relaxation corresponds to an optimization of the form in Eq. 10, with the marginal polytope $\mathcal{M}(\mathcal{G}'_x)$ replaced by $\mathcal{L}(\mathcal{G}'_x)$.

$\mathcal{G}'_x$ includes the following auxiliary variable nodes: *path variables* $\langle p_{ij} \rangle_{i=0,\ldots,n,j=1,\ldots,n}$, which indicate whether word j descends from i in the dependency tree, and *flow variables* $\langle f_a^k \rangle_{a \in A, k=1,\ldots,n}$, which evaluate to 1 iff arc a "carries flow" to k, i.e., iff there is a path from the root to k that passes through a. We need to seed these variables imposing

$$p_{0k} = p_{kk} = 1, \forall k, \quad f^h_{\langle h,m \rangle} = 0, \forall h, m; \tag{15}$$

i.e., any word descends from the root and from itself, and arcs leaving a word carry no flow to that

[6] Smith and Eisner (2008) also proposed other variants with more factors, which we omit for brevity.

word. This can be done with unary hard constraint factors. We then replace the TREE factor in Fig. 1 by the factors shown in Fig. 3:

- $O(n)$ XOR factors, each connecting all arc variables of the form $\{\langle h,m\rangle\}_{h=0,\ldots,n}$. These ensure that each word has exactly one parent. Each factor yields a local agreement constraint (see Table 1):
$$\textstyle\sum_{h=0}^{n} z_{\langle h,m\rangle} = 1, \quad m \in \{1,\ldots,n\} \qquad (16)$$
- $O(n^3)$ IMPLY factors, each expressing that if an arc carries flow, then that arc *must* be active. Such factors are OR factors with the first input negated, hence, the local agreement constraints are:
$$f_a^k \leq z_a, \quad a \in A, k \in \{1,\ldots,n\}. \qquad (17)$$
- $O(n^2)$ XOR-WITH-OUTPUT factors, which impose the constraint that each path variable p_{mk} is active if and only if exactly one incoming arc in $\{\langle h,m\rangle\}_{h=0,\ldots,n}$ carries flow to k. Such factors are XOR factors with the last input negated, and hence their local constraints are:
$$p_{mk} = \textstyle\sum_{h=0}^{n} f_{\langle h,m\rangle}^k, \quad m,k \in \{1,\ldots,n\} \quad (18)$$
- $O(n^2)$ XOR-WITH-OUTPUT factors to impose the constraint that words don't consume other words' commodities; i.e., if $h \neq k$ and $k \neq 0$, then there is a path from h to k iff exactly one outgoing arc in $\{\langle h,m\rangle\}_{m=1,\ldots,n}$ carries flow to k:
$$p_{hk} = \textstyle\sum_{m=1}^{n} f_{\langle h,m\rangle}^k, \ h,k \in \{0,\ldots,n\}, \ k \notin \{0,h\}. \qquad (19)$$

$\mathcal{L}(\mathcal{G}'_x)$ is thus defined by the constraints in Eq. 12 and 15–19. The approximate MAP problem, that replaces $\mathcal{M}(\mathcal{G}'_x)$ by $\mathcal{L}(\mathcal{G}'_x)$ in Eq. 10, thus becomes:

$$\begin{array}{ll} \max_{\mathbf{z},\mathbf{f},\mathbf{p}} & \boldsymbol{\theta}^\top \mathbf{F}(x)\mathbf{z} \\ \text{s.t.} & \text{Eqs. 12 and 15–19 are satisfied.} \end{array} \qquad (20)$$

This is exactly the LP relaxation considered by Martins et al. (2009) in their multi-commodity flow model, for the configuration with siblings and grandparent features.[7] They also considered a configuration with non-projectivity features—which fire if an arc is non-projective.[8] That configuration can also be obtained here if variables $\{n_{\langle h,m\rangle}\}$ are added to indicate non-projective arcs and OR-WITH-OUTPUT hard constraint factors are inserted to enforce $n_{\langle h,m\rangle} = z_{\langle h,m\rangle} \wedge \bigvee_{\min(h,m)<j<\min(h,m)} \neg p_{hj}$. Details are omitted for space.

In sum, although the approaches of Smith and Eisner (2008) and Martins et al. (2009) look very different, in reality both are variational approximations emanating from Prop. 1, respectively for marginal and MAP inference. However, they operate on distinct factor graphs, respectively Figs. 1 and 3.[9]

5 Online Learning

Our learning algorithm is presented in Alg. 1. It is a generalized online learner that tackles ℓ_2-regularized empirical risk minimization of the form

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \tfrac{\lambda}{2}\|\boldsymbol{\theta}\|^2 + \tfrac{1}{m}\textstyle\sum_{i=1}^{m} L(\boldsymbol{\theta}; x_i, \mathbf{y}_i), \qquad (21)$$

where each $\langle x_i, \mathbf{y}_i\rangle$ is a training example, $\lambda \geq 0$ is the regularization constant, and $L(\boldsymbol{\theta}; x, \mathbf{y})$ is a nonnegative convex loss. Examples include the logistic loss used in CRFs ($-\log \Pr_{\boldsymbol{\theta}}(\mathbf{y}|x)$) and the hinge loss of structured SVMs ($\max_{\mathbf{y}'\in\mathcal{Y}(x)} \boldsymbol{\theta}^\top(\boldsymbol{\phi}(x,\mathbf{y}') - \boldsymbol{\phi}(x,\mathbf{y})) + \ell(\mathbf{y}',\mathbf{y})$ for some cost function ℓ). These are both special cases of the family defined in Fig. 4, which also includes the structured perceptron's loss ($\beta \to \infty$, $\gamma = 0$) and the softmax-margin loss of Gimpel and Smith (2010; $\beta = \gamma = 1$).

Alg. 1 is closely related to stochastic or online gradient descent methods, but with the key advantage of not needing a learning rate hyperparameter. We sketch the derivation of Alg. 1; full details can be found in Martins et al. (2010a). On the tth round, one example $\langle x_t, \mathbf{y}_t\rangle$ is considered. We seek to solve

$$\begin{array}{ll} \min_{\boldsymbol{\theta},\xi} & \frac{\lambda m}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}_t\|^2 + \xi \\ \text{s.t.} & L(\boldsymbol{\theta}; x_t, \mathbf{y}_t) \leq \xi, \quad \xi \geq 0, \end{array} \qquad (23)$$

[7] To be precise, the constraints of Martins et al. (2009) are recovered after eliminating the path variables, via Eqs. 18–19.

[8] An arc $\langle h,m\rangle$ is non-projective if there is some word in its span not descending from h (Kahane et al., 1998).

[9] Given what was just exposed, it seems appealing to try max-product loopy BP on the factor graph of Fig. 1, or sum-product loopy BP on the one in Fig. 3. Both attempts present serious challenges: the former requires computing messages sent by the tree factor, which requires $O(n^2)$ calls to the Chu-Liu-Edmonds algorithm and hence $O(n^5)$ time. No obvious strategy seems to exist for simultaneous computation of all messages, unlike in the sum-product case. The latter is even more challenging, as standard sum-product loopy BP has serious issues in the factor graph of Fig. 3; we construct in Martins et al. (2010b) a simple example with a very poor Bethe approximation. This might be fixed by using other variants of sum-product BP, e.g., ones in which the entropy approximation is concave.

$$L_{\beta,\gamma}(\boldsymbol{\theta}; x, \mathbf{y}) \triangleq \tfrac{1}{\beta} \log \textstyle\sum_{\mathbf{y}' \in \mathcal{Y}(x)} \exp \left[\beta \Big(\boldsymbol{\theta}^\top \big(\boldsymbol{\phi}(x, \mathbf{y}') - \boldsymbol{\phi}(x, \mathbf{y}) \big) + \gamma \ell(\mathbf{y}', \mathbf{y}) \Big) \right] \tag{22}$$

Figure 4: A family of loss functions including as particular cases the ones used in CRFs, structured SVMs, and the structured perceptron. The hyperparameter β is the analogue of the inverse temperature in a Gibbs distribution, while γ scales the cost. For any choice of $\beta > 0$ and $\gamma \geq 0$, the resulting loss function is convex in $\boldsymbol{\theta}$, since, up to a scale factor, it is the composition of the (convex) log-sum-exp function with an affine map.

Algorithm 1 Aggressive Online Learning

```
1:  Input: ${\{\langle x_i, \mathbf{y}_i \rangle\}}_{i=1}^m$, $\lambda$, number of epochs $K$
2:  Initialize $\boldsymbol{\theta}_1 \leftarrow \mathbf{0}$; set $T = mK$
3:  for $t = 1$ to $T$ do
4:      Receive instance $\langle x_t, \mathbf{y}_t \rangle$ and set $\boldsymbol{\mu}_t = \boldsymbol{\chi}(\mathbf{y}_t)$
5:      Solve Eq. 24 to obtain $\bar{\boldsymbol{\mu}}_t$ and $L_{\beta,\gamma}(\boldsymbol{\theta}_t, x_t, \mathbf{y}_t)$
6:      Compute $\nabla L_{\beta,\gamma}(\boldsymbol{\theta}_t, x_t, \mathbf{y}_t) = \mathbf{F}(x_t)(\bar{\boldsymbol{\mu}}_t - \boldsymbol{\mu}_t)$
7:      Compute $\eta_t = \min \left\{ \frac{1}{\lambda m}, \frac{L_{\beta,\gamma}(\boldsymbol{\theta}_t; x_t, \mathbf{y}_t)}{\|\nabla L_{\beta,\gamma}(\boldsymbol{\theta}_t; x_t, \mathbf{y}_t)\|^2} \right\}$
8:      Return $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \nabla L_{\beta,\gamma}(\boldsymbol{\theta}_t; x_t, \mathbf{y}_t)$
9:  end for
10: Return the averaged model $\bar{\boldsymbol{\theta}} \leftarrow \frac{1}{T} \sum_{t=1}^T \boldsymbol{\theta}_t$.
```

which trades off conservativeness (stay close to the most recent solution $\boldsymbol{\theta}_t$) and correctness (keep the loss small). Alg. 1's lines 7–8 are the result of taking the first-order Taylor approximation of L around $\boldsymbol{\theta}_t$, which yields the lower bound $L(\boldsymbol{\theta}; x_t, \mathbf{y}_t) \geq L(\boldsymbol{\theta}_t; x_t, \mathbf{y}_t) + (\boldsymbol{\theta} - \boldsymbol{\theta}_t)^\top \nabla L(\boldsymbol{\theta}_t; x_t, \mathbf{y}_t)$, and plugging that linear approximation into the constraint of Eq. 23, which gives a simple Euclidean projection problem (with slack) with a closed-form solution.

The online updating requires evaluating the loss and computing its gradient. Both quantities can be computed using the variational expression in Prop. 1, for any loss $L_{\beta,\gamma}(\boldsymbol{\theta}; x, \mathbf{y})$ in Fig. 4.[10] Our only assumption is that the cost function $\ell(\mathbf{y}', \mathbf{y})$ can be written as a sum over factor-local costs; letting $\boldsymbol{\mu} = \boldsymbol{\chi}(\mathbf{y})$ and $\boldsymbol{\mu}' = \boldsymbol{\chi}(\mathbf{y}')$, this implies $\ell(\mathbf{y}', \mathbf{y}) = \mathbf{p}^\top \boldsymbol{\mu}' + q$ for some $\mathbf{p}$ and q which are constant with respect to $\boldsymbol{\mu}'$.[11] Under this assumption, $L_{\beta,\gamma}(\boldsymbol{\theta}; x, \mathbf{y})$ becomes expressible in terms of the log-partition function of a distribution whose log-potentials are set to $\beta(\mathbf{F}(x)^\top \boldsymbol{\theta} + \gamma \mathbf{p})$. From Eq. 9 and after some algebra, we finally obtain $L_{\beta,\gamma}(\boldsymbol{\theta}; x, \mathbf{y}) =$

$$\max_{\boldsymbol{\mu}' \in \mathcal{M}(\mathcal{G}_x)} \boldsymbol{\theta}^\top \mathbf{F}(x)(\boldsymbol{\mu}' - \boldsymbol{\mu}) + \frac{1}{\beta} H(\boldsymbol{\mu}') + \gamma(\mathbf{p}^\top \boldsymbol{\mu}' + q). \tag{24}$$

Let $\bar{\boldsymbol{\mu}}$ be a maximizer in Eq. 24; from the second statement of Prop. 1 we obtain $\nabla L_{\beta,\gamma}(\boldsymbol{\theta}; x, \mathbf{y}) = \mathbf{F}(x)(\bar{\boldsymbol{\mu}} - \boldsymbol{\mu})$. When the inference problem in Eq. 24 is intractable, approximate message-passing algorithms like loopy BP still allow us to obtain approximations of the loss $L_{\beta,\gamma}$ and its gradient.

For the hinge loss, we arrive precisely at the max-loss variant of 1-best MIRA (Crammer et al., 2006). For the logistic loss, we arrive at a new online learning algorithm for CRFs that resembles stochastic gradient descent but with an automatic step size that follows from our variational representation.

Unsupported Features. As datasets grow, so do the sets of features, creating further computational challenges. Often only "supported" features—those observed in the training data—are included, and even those are commonly eliminated when their frequencies fall below a threshold. Important information may be lost as a result of these expedient choices. Formally, the *supported feature set* is $\mathcal{F}_{\text{supp}} \triangleq \bigcup_{i=1}^m \operatorname{supp} \boldsymbol{\phi}(x_i, \mathbf{y}_i)$, where $\operatorname{supp} \mathbf{u} \triangleq \{j \mid u_j \neq 0\}$ denotes the *support* of vector $\mathbf{u}$. $\mathcal{F}_{\text{supp}}$ is a subset of the *complete* feature set, comprised of those features that occur in some candidate output, $\mathcal{F}_{\text{comp}} \triangleq \bigcup_{i=1}^m \bigcup_{\mathbf{y}'_i \in \mathcal{Y}(x_i)} \operatorname{supp} \boldsymbol{\phi}(x_i, \mathbf{y}'_i)$. Features in $\mathcal{F}_{\text{comp}} \setminus \mathcal{F}_{\text{supp}}$ are called *unsupported*.

Sha and Pereira (2003) have shown that training a CRF-based shallow parser with the complete feature set may improve performance (over the supported one), at the cost of 4.6 times more features. Dependency parsing has a much higher ratio (around 20 for bilexical word-word features, as estimated in the Penn Treebank), due to the quadratic or faster growth of the number of parts, of which only a few are active in a legal output. We propose a simple strategy for handling $\mathcal{F}_{\text{comp}}$ efficiently, which can be applied for those losses in Fig. 4 where $\beta = \infty$. (e.g., the structured SVM and perceptron). Our procedure is the following: keep an active set $\mathcal{F}$ contain-

[10] Our description also applies to the (non-differentiable) hinge loss case, when $\beta \rightarrow \infty$, if we replace all instances of "the gradient" in the text by "a subgradient."

[11] For the Hamming cost, this holds with $\mathbf{p} = \mathbf{1} - 2\boldsymbol{\mu}$ and $q = \mathbf{1}^\top \boldsymbol{\mu}$. See Taskar et al. (2006) for other examples.

	CRF (Turbo Pars. #1)		SVM (Turbo Pars. #2)		SVM (Turbo #2)		
	Arc-Fact.	Sec. Ord.	Arc-Fact.	Sec. Ord.	$\lvert\mathcal{F}\rvert$	$\frac{\lvert\mathcal{F}\rvert}{\lvert\mathcal{F}_{\mathrm{supp}}\rvert}$	+nonproj., compl.
Arabic	78.28	79.12	79.04	79.42	6,643,191	2.8	80.02 (-0.14)
Bulgarian	91.02	91.78	90.84	92.30	13,018,431	2.1	92.88 (+0.34) (†)
Chinese	90.58	90.87	91.09	91.77	28,271,086	2.1	91.89 (+0.26)
Czech	86.18	87.72	86.78	88.52	83,264,645	2.3	88.78 (+0.44) (†)
Danish	89.58	90.08	89.78	90.78	7,900,061	2.3	91.50 (+0.68)
Dutch	82.91	84.31	82.73	84.17	15,652,800	2.1	84.91 (-0.08)
German	89.34	90.58	89.04	91.19	49,934,403	2.5	91.49 (+0.32) (†)
Japanese	92.90	93.22	93.18	93.38	4,256,857	2.2	93.42 (+0.32)
Portuguese	90.64	91.00	90.56	91.50	16,067,150	2.1	91.87 (-0.04)
Slovene	83.03	83.17	83.49	84.35	4,603,295	2.7	85.53 (+0.80)
Spanish	83.83	85.07	84.19	85.95	11,629,964	2.6	87.04 (+0.50) (†)
Swedish	87.81	89.01	88.55	88.99	18,374,160	2.8	89.80 (+0.42)
Turkish	76.86	76.28	74.79	76.10	6,688,373	2.2	76.62 (+0.62)
English Non-Proj.	90.15	91.08	90.66	91.79	57,615,709	2.5	92.13 (+0.12)
English Proj.	91.23	91.94	91.65	92.91	55,247,093	2.4	93.26 (+0.41) (†)

Table 2: Unlabeled attachment scores, ignoring punctuation. The leftmost columns show the performance of arc-factored and second-order models for the CRF and SVM losses, after 10 epochs with $1/(\lambda m) = 0.001$ (tuned on the English Non-Proj. dev.-set). The rightmost columns refer to a model to which non-projectivity features were added, trained under the SVM loss, that handles the *complete* feature set. Shown is the total number of features instantiated, the multiplicative factor w.r.t. the number of supported features, and the accuracies (in parenthesis, we display the difference w.r.t. a model trained with the supported features only). Entries marked with † are the highest reported in the literature, to the best of our knowledge, beating (sometimes slightly) McDonald et al. (2006), Martins et al. (2008), Martins et al. (2009), and, in the case of English Proj., also the third-order parser of Koo and Collins (2010), which achieves 93.04% on that dataset (their experiments in Czech are not comparable, since the datasets are different).

ing all features that have been instantiated in Alg. 1. At each round, run lines 4–5 as usual, using only features in $\mathcal{F}$. Since the other features have not been used before, they have a zero weight, hence can be ignored. When $\beta = \infty$, the variational problem in Eq. 24 consists of a MAP computation and the solution corresponds to one output $\hat{\mathbf{y}}_t \in \mathcal{Y}(x_t)$. Only the parts that are active in $\hat{\mathbf{y}}_t$ but not in $\mathbf{y}_t$, or vice-versa, will have features that might receive a nonzero update. Those parts are reexamined for new features and the active set $\mathcal{F}$ is updated accordingly.

6 Experiments

We trained non-projective dependency parsers for 14 languages, using datasets from the CoNLL-X shared task (Buchholz and Marsi, 2006) and two datasets for English: one from the CoNLL-2008 shared task (Surdeanu et al., 2008), which contains non-projective arcs, and another derived from the Penn Treebank applying the standard head rules of Yamada and Matsumoto (2003), in which all parse trees are projective.[12] We implemented Alg. 1, which handles any loss function $L_{\beta,\gamma}$.[13] When $\beta < \infty$, Turbo Parser #1 and the loopy BP algorithm of Smith and Eisner (2008) is used; otherwise, Turbo Parser #2 is used and the LP relaxation is solved with CPLEX. In both cases, we employed the same pruning strategy as Martins et al. (2009).

Two different feature configurations were first tried: an arc-factored model and a model with second-order features (siblings and grandparents). We used the same arc-factored features as McDonald et al. (2005) and second-order features that conjoin words and lemmas (at most two), parts-of-speech tags, and (if available) morphological information; this was the same set of features as in Martins et al. (2009). Table 2 shows the results obtained in both configurations, for CRF and SVM loss functions. While in the arc-factored case performance is similar, in second-order models there seems to be a consistent gain when the SVM loss is used. There are two possible reasons: first, SVMs take the cost function into consideration; second, Turbo Parser #2 is less approximate than Turbo Parser #1, since only the marginal polytope is approximated (the entropy function is not involved).

[12] We used the provided train/test splits for all datasets. For English, we used the standard test partitions (section 23 of the Wall Street Journal). We did not exploit the fact that some datasets only contain projective trees and have unique roots.

[13] The code is available at `http://www.ark.cs.cmu.edu/TurboParser`.

β	1	1	1	1	3	5	∞
γ	0 (CRF)	1	3	5	1	1	1 (SVM)
ARC-F.	90.15	90.41	90.38	90.53	90.80	90.83	90.66
2 ORD.	91.08	91.85	91.89	91.51	92.04	91.98	91.79

Table 3: Varying β and γ: neither the CRF nor the SVM is optimal. Results are UAS on the English Non-Projective dataset, with λ tuned with dev.-set validation.

The loopy BP algorithm managed to converge for nearly all sentences (with message damping). The last three columns show the beneficial effect of unsupported features for the SVM case (with a more powerful model with non-projectivity features). For most languages, unsupported features convey helpful information, which can be used with little extra cost (on average, 2.5 times more features are instantiated). A combination of the techniques discussed here yields parsers that are in line with very strong competitors—for example, the parser of Koo and Collins (2010), which is exact, third-order, and constrains the outputs to be projective, does not outperform ours on the projective English dataset.[14]

Finally, Table 3 shows results obtained for different settings of β and γ. Interestingly, we observe that higher scores are obtained for loss functions that are "between" SVMs and CRFs.

7 Related Work

There has been recent work studying efficient computation of messages in combinatorial factors: bipartite matchings (Duchi et al., 2007), projective and non-projective arborescences (Smith and Eisner, 2008), as well as high order factors with count-based potentials (Tarlow et al., 2010), among others. Some of our combinatorial factors (OR, OR-WITH-OUTPUT) and the analogous entropy computations were never considered, to the best of our knowledge.

Prop. 1 appears in Wainwright and Jordan (2008) for canonical overcomplete models; we adapt it here for models with shared features. We rely on the variational interpretation of loopy BP, due to Yedidia et al. (2001), to derive the objective being optimized by Smith and Eisner's loopy BP parser.

Independently of our work, Koo et al. (2010) recently proposed an efficient dual decomposition method to solve an LP problem similar (but not equal) to the one in Eq. 20,[15] with excellent parsing performance. Their parser is also an instance of a turbo parser since it relies on a local approximation of a marginal polytope. While one can also use dual decomposition to address our MAP problem, the fact that our model does not decompose as nicely as the one in Koo et al. (2010) would likely result in slower convergence.

8 Conclusion

We presented a unified view of two recent approximate dependency parsers, by stating their underlying factor graphs and by deriving the variational problems that they address. We introduced new hard constraint factors, along with formulae for their messages, local belief constraints, and entropies. We provided an aggressive online algorithm for training the models with a broad family of losses.

There are several possible directions for future work. Recent progress in message-passing algorithms yield "convexified" Bethe approximations that can be used for marginal inference (Wainwright et al., 2005), and provably convergent max-product variants that solve the relaxed LP (Globerson and Jaakkola, 2008). Other parsing formalisms can be handled with the inventory of factors shown here—among them, phrase-structure parsing.

Acknowledgments

The authors would like to thank the reviewers for their comments, and Kevin Gimpel, David Smith, David Sontag, and Terry Koo for helpful discussions. A. M. was supported by a grant from FCT/ICTI through the CMU-Portugal Program, and also by Priberam Informática. N. S. was supported in part by Qatar NRF NPRP-08-485-1-083. E. X. was supported by AFOSR FA9550010247, ONR N000140910758, NSF CAREER DBI-0546594, NSF IIS-0713379, and an Alfred P. Sloan Fellowship. M. F. and P. A. were supported by the FET programme (EU FP7), under the SIMBAD project (contract 213250).

[14] This might be due to the fact that Koo and Collins (2010) trained with the perceptron algorithm and did not use unsupported features. Experiments plugging the perceptron loss ($\beta \to \infty, \gamma \to 0$) into Alg. 1 yielded worse performance than with the hinge loss.

[15] The difference is that the model of Koo et al. (2010) includes features that depend on *consecutive* siblings—making it decompose into subproblems amenable to dynamic programming—while we have factors for *all pairs* of siblings.

References

C. Berrou, A. Glavieux, and P. Thitimajshima. 1993. Near Shannon limit error-correcting coding and decoding. In *Proc. of ICC*, volume 93, pages 1064–1070.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *CoNLL*.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. 2006. Online passive-aggressive algorithms. *JMLR*, 7:551–585.

J. Duchi, D. Tarlow, G. Elidan, and D. Koller. 2007. Using combinatorial optimization within max-product belief propagation. *NIPS*, 19.

J. R. Finkel, A. Kleeman, and C. D. Manning. 2008. Efficient, feature-based, conditional random field parsing. *Proc. of ACL*.

K. Gimpel and N. A. Smith. 2010. Softmax-margin crfs: Training log-linear models with loss functions. In *Proc. of NAACL*.

A. Globerson and T. Jaakkola. 2008. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. *NIPS*, 20.

L. Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proc. of ACL*.

S. Kahane, A. Nasr, and O. Rambow. 1998. Pseudo-projectivity: a polynomially parsable non-projective dependency grammar. In *Proc. of COLING*.

T. Koo and M. Collins. 2010. Efficient third-order dependency parsers. In *Proc. of ACL*.

T. Koo, A. Globerson, X. Carreras, and M. Collins. 2007. Structured prediction models via the matrix-tree theorem. In *Proc. of EMNLP*.

T. Koo, A. M. Rush, M. Collins, T. Jaakkola, and D. Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proc. of EMNLP*.

F. R. Kschischang, B. J. Frey, and H. A. Loeliger. 2001. Factor graphs and the sum-product algorithm. *IEEE Trans. Inf. Theory*, 47(2):498–519.

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

A. F. T. Martins, D. Das, N. A. Smith, and E. P. Xing. 2008. Stacking dependency parsers. In *EMNLP*.

A. F. T. Martins, N. A. Smith, and E. P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proc. of ACL-IJCNLP*.

A. F. T. Martins, K. Gimpel, N. A. Smith, E. P. Xing, P. M. Q. Aguiar, and M. A. T. Figueiredo. 2010a. Learning structured classifiers with dual coordinate descent. Technical Report CMU-ML-10-109.

A. F. T. Martins, N. A. Smith, E. P. Xing, P. M. Q. Aguiar, and M. A. T. Figueiredo. 2010b. Turbo parsers: Dependency parsing by approximate variational inference (extended version).

A. McCallum, K. Schultz, and S. Singh. 2009. Factorie: Probabilistic programming via imperatively defined factor graphs. In *NIPS*.

R. T. McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of HLT-EMNLP*.

R. McDonald, K. Lerman, and F. Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proc. of CoNLL*.

R. J. McEliece, D. J. C. MacKay, and J. F. Cheng. 1998. Turbo decoding as an instance of Pearl's "belief propagation" algorithm. *IEEE Journal on Selected Areas in Communications*, 16(2).

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

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

D. A. Smith and J. Eisner. 2008. Dependency parsing by belief propagation. In *Proc. of EMNLP*.

D. A. Smith and N. A. Smith. 2007. Probabilistic models of nonprojective dependency trees. In *EMNLP*.

M. Surdeanu, R. Johansson, A. Meyers, L. Màrquez, and J. Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. *CoNLL*.

C. Sutton, A. McCallum, and K. Rohanimanesh. 2007. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *JMLR*, 8:693–723.

R. E. Tarjan. 1977. Finding optimum branchings. *Networks*, 7(1):25–36.

D. Tarlow, I. E. Givoni, and R. S. Zemel. 2010. HOP-MAP: Efficient message passing with high order potentials. In *Proc. of AISTATS*.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin Markov networks. In *NIPS*.

B. Taskar, S. Lacoste-Julien, and M. I. Jordan. 2006. Structured prediction, dual extragradient and Bregman projections. *JMLR*, 7:1627–1653.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proc. of ICML*.

M. J. Wainwright and M. I. Jordan. 2008. *Graphical Models, Exponential Families, and Variational Inference.* Now Publishers.

M. J. Wainwright, T.S. Jaakkola, and A.S. Willsky. 2005. A new class of upper bounds on the log partition function. *IEEE Trans. Inf. Theory*, 51(7):2313–2335.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. of IWPT*.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. 2001. Generalized belief propagation. In *NIPS*.

Holistic Sentiment Analysis Across Languages: Multilingual Supervised Latent Dirichlet Allocation

Jordan Boyd-Graber
UMD iSchool
and UMIACS
University of Maryland
College Park, MD
jbg@umiacs.umd.edu

Philip Resnik
Department of Linguistics
and UMIACS
University of Maryland
College Park, MD
resnik@umd.edu

Abstract

In this paper, we develop multilingual supervised latent Dirichlet allocation (MLSLDA), a probabilistic generative model that allows insights gleaned from one language's data to inform how the model captures properties of other languages. MLSLDA accomplishes this by jointly modeling two aspects of text: how multilingual concepts are clustered into thematically coherent topics and how topics associated with text connect to an observed regression variable (such as ratings on a sentiment scale). Concepts are represented in a general hierarchical framework that is flexible enough to express semantic ontologies, dictionaries, clustering constraints, and, as a special, degenerate case, conventional topic models. Both the topics and the regression are discovered via posterior inference from corpora. We show MLSLDA can build topics that are consistent across languages, discover sensible bilingual lexical correspondences, and leverage multilingual corpora to better predict sentiment.

Sentiment analysis (Pang and Lee, 2008) offers the promise of automatically discerning how people feel about a product, person, organization, or issue based on what they write online, which is potentially of great value to businesses and other organizations. However, the vast majority of sentiment resources and algorithms are limited to a single language, usually English (Wilson, 2008; Baccianella and Sebastiani, 2010). Since no single language captures a majority of the content online, adopting such a limited approach in an increasingly global community risks missing important details and trends that might only be available when text in multiple languages is taken into account.

Up to this point, multiple languages have been addressed in sentiment analysis primarily by transferring knowledge from a resource-rich language to a less rich language (Banea et al., 2008), or by ignoring differences in languages via translation into English (Denecke, 2008). These approaches are limited to a view of sentiment that takes place through an English-centric lens, and they ignore the potential to share information between languages. Ideally, learning sentiment cues holistically, *across* languages, would result in a richer and more globally consistent picture.

In this paper, we introduce Multilingual Supervised Latent Dirichlet Allocation (MLSLDA), a model for sentiment analysis on a multilingual corpus. MLSLDA discovers a consistent, unified picture of sentiment across multiple languages by learning "topics," probabilistic partitions of the vocabulary that are consistent in terms of both meaning and relevance to observed sentiment. Our approach makes few assumptions about available resources, requiring neither parallel corpora nor machine translation.

The rest of the paper proceeds as follows. In Section 1, we describe the probabilistic tools that we use to create consistent topics bridging across languages and the MLSLDA model. In Section 2, we present the inference process. We discuss our set of semantic bridges between languages in Section 3, and our experiments in Section 4 demonstrate that this approach functions as an effective multilingual topic model, discovers sentiment-biased topics, and uses multilingual corpora to make better sentiment predictions across languages. Sections 5 and 6 discuss related research and discusses future work, respectively.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 45–55,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

1 Predictions from Multilingual Topics

As its name suggests, MLSLDA is an extension of Latent Dirichlet allocation (LDA) (Blei et al., 2003), a modeling approach that takes a corpus of unannotated documents as input and produces two outputs, a set of "topics" and assignments of documents to topics. Both the topics and the assignments are probabilistic: a topic is represented as a probability distribution over words in the corpus, and each document is assigned a probability distribution over all the topics. Topic models built on the foundations of LDA are appealing for sentiment analysis because the learned topics can cluster together sentiment-bearing words, and because topic distributions are a parsimonious way to represent a document.[1]

LDA has been used to discover latent structure in text (e.g. for discourse segmentation (Purver et al., 2006) and authorship (Rosen-Zvi et al., 2004)). MLSLDA extends the approach by ensuring that this latent structure — the underlying topics — is consistent across languages. We discuss multilingual topic modeling in Section 1.1, and in Section 1.2 we show how this enables supervised regression regardless of a document's language.

1.1 Capturing Semantic Correlations

Topic models posit a straightforward generative process that creates an observed corpus. For each document d, some distribution θ_d over unobserved topics is chosen. Then, for each word position in the document, a topic z is selected. Finally, the word for that position is generated by selecting from the topic indexed by z. (Recall that in LDA, a "topic" is a distribution over words).

In monolingual topic models, the topic distribution is usually drawn from a Dirichlet distribution. Using Dirichlet distributions makes it easy to specify sparse priors, and it also simplifies posterior inference because Dirichlet distributions are conjugate to multinomial distributions. However, drawing topics from Dirichlet distributions will not suffice if our vocabulary includes multiple languages. If we are working with English, German, and Chinese at the same time, a Dirichlet prior has no way to favor distributions z such that $p(\text{good}|z)$, $p(\text{gut}|z)$, and $p(\text{hǎo}|z)$ all tend to be high at the same time, or low at the same time. More generally, the structure of our model must encourage topics to be consistent across languages, and Dirichlet distributions cannot encode correlations between elements.

One possible solution to this problem is to use the multivariate normal distribution, which can produce correlated multinomials (Blei and Lafferty, 2005), in place of the Dirichlet distribution. This has been done successfully in multilingual settings (Cohen and Smith, 2009). However, such models complicate inference by not being conjugate.

Instead, we appeal to tree-based extensions of the Dirichlet distribution, which has been used to induce correlation in semantic ontologies (Boyd-Graber et al., 2007) and to encode clustering constraints (Andrzejewski et al., 2009). The key idea in this approach is to assume the vocabularies of all languages are organized according to some shared semantic structure that can be represented as a tree. For concreteness in this section, we will use WordNet (Miller, 1990) as the representation of this multilingual semantic bridge, since it is well known, offers convenient and intuitive terminology, and demonstrates the full flexibility of our approach. However, the model we describe generalizes to any tree-structured representation of multilingual knowledge; we discuss some alternatives in Section 3.

WordNet organizes a vocabulary into a rooted, directed acyclic graph of nodes called synsets, short for "synonym sets." A synset is a child of another synset if it satisfies a hyponomy relationship; each child "is a" more specific instantiation of its parent concept (thus, hyponomy is often called an "isa" relationship). For example, a "dog" is a "canine" is an "animal" is a "living thing," etc. As an approximation, it is not unreasonable to assume that WordNet's structure of meaning is language independent, i.e. the concept encoded by a synset can be realized using terms in different languages that share the same meaning. In practice, this organization has been used to create many alignments of international WordNets to the original English WordNet (Ordan and Wintner, 2007; Sagot and Fišer, 2008; Isahara et al., 2008).

Using the structure of WordNet, we can now describe a generative process that produces a distribution over a multilingual vocabulary, which encourages correlations between words with similar mean-

[1] The latter property has also made LDA popular for information retrieval (Wei and Croft, 2006)).

ings regardless of what language each word is in. For each synset h, we create a multilingual word distribution for that synset as follows:

1. Draw transition probabilities $\beta_h \sim \text{Dir}(\tau_h)$
2. Draw stop probabilities $\omega_h \sim \text{Dir}(\kappa_h)$
3. For each language l, draw emission probabilities for that synset $\phi_{h,l} \sim \text{Dir}(\pi_{h,l})$.

For conciseness in the rest of the paper, we will refer to this generative process as *multilingual Dirichlet hierarchy*, or MULTDIRHIER$(\boldsymbol{\tau}, \boldsymbol{\kappa}, \boldsymbol{\pi})$.[2] Each observed token can be viewed as the end result of a sequence of visited synsets λ. At each node in the tree, the path can end at node i with probability $\omega_{i,1}$, or it can continue to a child synset with probability $\omega_{i,0}$. If the path continues to another child synset, it visits child j with probability $\beta_{i,j}$. If the path ends at a synset, it generates word k with probability $\phi_{i,l,k}$.[3] The probability of a word being emitted from a path with visited synsets r and final synset h in language l is therefore

$$p(w, \lambda = r, h | l, \boldsymbol{\beta}, \boldsymbol{\omega}, \boldsymbol{\phi}) = \left(\prod_{(i,j) \in r} \beta_{i,j} \omega_{i,0} \right) (1 - \omega_{h,1}) \phi_{h,l,w}. \quad (1)$$

Note that the stop probability ω_h is independent of language, but the emission $\phi_{h,l}$ is dependent on the language. This is done to prevent the following scenario: while synset A is highly probable in a topic and words in language 1 attached to that synset have high probability, words in language 2 have low probability. If this could happen for many synsets in a topic, an entire language would be effectively silenced, which would lead to inconsistent topics (e.g. Topic 1 is about baseball in English and about travel in German). Separating path from emission helps ensure that topics are consistent across languages.

Having defined topic distributions in a way that can preserve cross-language correspondences, we now use this distribution within a larger model that can discover cross-language patterns of use that predict sentiment.

1.2 The MLSLDA Model

We will view sentiment analysis as a regression problem: given an input document, we want to predict a real-valued observation y that represents the sentiment of a document. Specifically, we build on supervised latent Dirichlet allocation (SLDA, (Blei and McAuliffe, 2007)), which makes predictions based on the topics expressed in a document; this can be thought of projecting the words in a document to low dimensional space of dimension equal to the number of topics. Blei et al. showed that using this latent topic structure can offer improved predictions over regressions based on words alone, and the approach fits well with our current goals, since word-level cues are unlikely to be identical across languages. In addition to text, SLDA has been successfully applied to other domains such as social networks (Chang and Blei, 2009) and image classification (Wang et al., 2009). The key innovation in this paper is to extend SLDA by creating topics that are globally consistent across languages, using the bridging approach above.

We express our model in the form of a probabilistic generative latent-variable model that generates documents in multiple languages *and* assigns a real-valued score to each document. The score comes from a normal distribution whose sum is the dot product between a regression parameter η that encodes the influence of each topic on the observation and a variance σ^2. With this model in hand, we use statistical inference to determine the distribution over latent variables that, given the model, best explains observed data.

The generative model is as follows:

1. For each topic $i = 1 \dots K$, draw a topic distribution $\{\boldsymbol{\beta}_i, \boldsymbol{\omega}_i, \boldsymbol{\phi}_i\}$ from MULTDIRHIER$(\boldsymbol{\tau}, \boldsymbol{\kappa}, \boldsymbol{\pi})$.
2. For each document $d = 1 \dots M$ with language l_d:
   (a) Choose a distribution over topics $\theta_d \sim \text{Dir}(\alpha)$.

[2] Variables τ_h, $\pi_{h,l}$, and κ_h are hyperparameters. Their mean is fixed, but their magnitude is sampled during inference (i.e. $\frac{\tau_{h,i}}{\sum_k \tau_{h,k}}$ is constant, but $\tau_{h,i}$ is not). For the bushier bridges, (e.g. dictionary and flat), their mean is uniform. For GermaNet, we took frequencies from two balanced corpora of German and English: the British National Corpus (University of Oxford, 2006) and the Kern Corpus of the Digitales Wörterbuch der Deutschen Sprache des 20. Jahrhunderts project (Geyken, 2007). We took these frequencies and propagated them through the multilingual hierarchy, following LDAWN's (Boyd-Graber et al., 2007) formulation of information content (Resnik, 1995) as a Bayesian prior. The variance of the priors was initialized to be 1.0, but could be sampled during inference.

[3] Note that the language and word are taken as given, but the path through the semantic hierarchy is a latent random variable.

    (b) For each word in the document $n = 1 \ldots N_d$, choose a topic assignment $z_{d,n} \sim \mathrm{Mult}\left(\theta_d\right)$ and a path $\lambda_{d,n}$ ending at word $w_{d,n}$ according to Equation 1 using $\{\boldsymbol{\beta}_{z_{d,n}}, \boldsymbol{\omega}_{z_{d,n}}, \boldsymbol{\phi}_{z_{d,n}}\}$.

3. Choose a response variable from $y \sim \mathrm{Norm}\left(\eta^\top \bar{z}, \sigma^2\right)$, where $\bar{z_d} \equiv \frac{1}{N}\sum_{n=1}^{N} z_{d,n}$.

Crucially, note that the topics are not independent of the sentiment task; the regression encourages terms with similar effects on the observation y to be in the same topic. The consistency of topics described above allows the same regression to be done for the entire corpus regardless of the language of the underlying document.

2 Inference

Finding the model parameters most likely to explain the data is a problem of statistical inference. We employ stochastic EM (Diebolt and Ip, 1996), using a Gibbs sampler for the E-step to assign words to paths and topics. After randomly initializing the topics, we alternate between sampling the topic and path of a word $(z_{d,n}, \lambda_{d,n})$ and finding the regression parameters η that maximize the likelihood. We jointly sample the topic and path conditioning on all of the other path and document assignments in the corpus, selecting a path and topic with probability

$$\begin{aligned} p(z_n = k, \lambda_n = r | \boldsymbol{z}_{-n}, \boldsymbol{\lambda}_{-n}, w_n, \eta, \sigma, \Theta) = \\ p(y_d | \boldsymbol{z}, \eta, \sigma) p(\lambda_n = r | z_n = k, \boldsymbol{\lambda}_{-n}, w_n, \boldsymbol{\tau}, \boldsymbol{\kappa}, \boldsymbol{\pi}) \\ p(z_n = k | \boldsymbol{z}_{-n}, \alpha). \qquad (2) \end{aligned}$$

Each of these three terms reflects a different influence on the topics from the vocabulary structure, the document's topics, and the response variable. In the next paragraphs, we will expand each of them to derive the full conditional topic distribution.

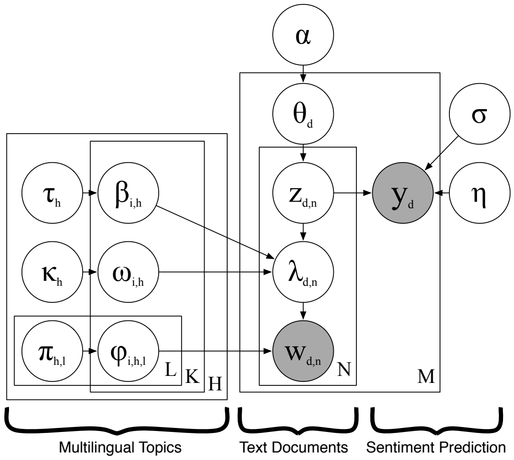


Figure 1: Graphical model representing MLSLDA. Shaded nodes represent observations, plates denote replication, and lines show probabilistic dependencies.

As discussed in Section 1.1, the structure of the topic distribution encourages terms with the same meaning to be in the same topic, even across languages. During inference, we marginalize over possible multinomial distributions β, ω, and ϕ, using the observed transitions from i to j in topic k; $T_{k,i,j}$, stop counts in synset i in topic k, $O_{k,i,0}$; continue counts in synsets i in topic k, $O_{k,i,1}$; and emission counts in synset i in language l in topic k, $F_{k,i,l}$. The probability of taking a path r is then

$$p(\lambda_n = r | z_n = k, \boldsymbol{\lambda}_{-n}) = \underbrace{\prod_{(i,j)\in r}\left(\frac{B_{k,i,j}+\tau_{i,j}}{\sum_{j'} B_{k,i,j'}+\tau_{i,j}}\frac{O_{k,i,1}+\omega_i}{\sum_{s\in 0,1} O_{k,i,s}+\omega_{i,s}}\right)}_{\text{Transition}} \underbrace{\frac{O_{k,r_{end},0}+\omega_{r_{end}}}{\sum_{s\in 0,1} O_{k,r_{end},s}+\omega_{r_{end},s}}\frac{F_{k,r_{end},w_n}+\pi_{r_{end},l}}{\sum_{w'} F_{r_{end},w'}+\pi_{r_{end},w'}}}_{\text{Emission}}. \qquad (3)$$

Equation 3 reflects the *multilingual* aspect of this model. The conditional topic distribution for SLDA (Blei and McAuliffe, 2007) replaces this term with the standard Multinomial-Dirichlet. However, we believe this is the first published SLDA-style model using MCMC inference, as prior work has used variational inference (Blei and McAuliffe, 2007; Chang and Blei, 2009; Wang et al., 2009).

Because the observed response variable depends on the topic assignments of a document, the conditional topic distribution is shifted toward topics that explain the observed response. Topics that move the predicted response $\hat{y}_d$ toward the true y_d will be favored. We drop terms that are constant across all topics for the effect of the response variable,

$$p(y_d|\boldsymbol{z}, \eta, \sigma) \propto \underbrace{\exp\left[\frac{1}{\sigma^2}\left(y_d - \frac{\sum_{k'} N_{d,k'}\eta_{k'}}{\sum_{k'} N_{d,k'}}\right)\frac{\eta_{z_k}}{\sum_{k'} N_{d,k'}}\right]}_{\text{Other words' influence}} \underbrace{\exp\left[\frac{-\eta_{z_k}^2}{2\sigma^2 \sum_{k'} N_{d,k'}^2}\right]}_{\text{This word's influence}}. \quad (4)$$

The above equation represents the *supervised* aspect of the model, which is inherited from SLDA.

Finally, there is the effect of the topics already assigned to a document; the conditional distribution favors topics already assigned in a document,

$$p(z_n = k|\boldsymbol{z}_{-n}, \alpha) = \frac{T_{d,k} + \alpha_k}{\sum_{k'} T_{d,k'} + \alpha_{k'}}. \quad (5)$$

This term represents the *document* focus of this model; it is present in all Gibbs sampling inference schemes for LDA (Griffiths and Steyvers, 2004).

Multiplying together Equations 3, 4, and 5 allows us to sample a topic using the conditional distribution from Equation 2, based on the topic and path of the other words in all languages. After sampling the path and topic for each word in a document, we then find new regression parameters η that maximize the likelihood conditioned on the current state of the sampler. This is simply a least squares regression using the topic assignments $\bar{z}_d$ to predict y_d.

Prediction on documents for which we don't have an observed y_d is equivalent to marginalizing over y_d and sampling topics for the document from Equations 3 and 5. The prediction for y_d is then the dot product of η and the empirical topic distribution $\bar{z}_d$.

We initially optimized all hyperparameters using slice sampling. However, we found that the regression variance σ^2 was not stable. Optimizing σ^2 seems to balance between modeling the language in the documents and the prediction, and thus is sensitive to documents' length. Given this sensitivity, we did not optimize σ^2 for our prediction experiments in Section 4, but instead kept it fixed at 0.25. We leave optimizing this variable, either through cross validation or adapting the model, to future work.

3 Bridges Across Languages

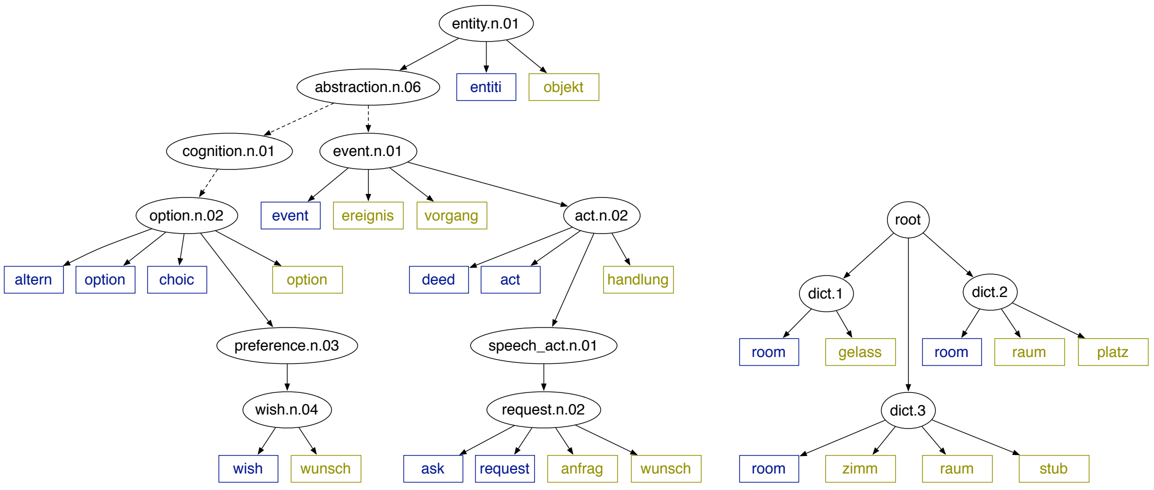
In Section 1.1, we described connections across languages as offered by semantic networks in a general way, using WordNet as an example. In this section, we provide more specifics, as well as alternative ways of building semantic connections across languages.

Flat First, we can consider a degenerate mapping that is nearly equivalent to running SLDA independently across multiple languages, relating topics only based on the impact on the response variable. Consider a degenerate tree with only one node, with all words in all languages associated with that node. This is consistent with our model, but there is really no shared semantic space, as all emitted words must come from this degenerate "synset" and the model only represents the output distribution for this single node.

WordNet We took the alignment of GermaNet to WordNet 1.6 (Kunze and Lemnitzer, 2002) and removed all synsets that were had no mapped German words. Any German synsets that did not have English translations had their words mapped to the lowest extant English hypernym (e.g. "beinbruch," a broken leg, was mapped to "fracture"). We stemmed all words to account for inflected forms not being present (Porter and Boulton, 1970). An example of the paths for the German word "wunsch" (wish, request) is shown in Figure 2(a).

Dictionaries A dictionary can be viewed as a many to many mapping, where each entry e_i maps one or more words in one language $\boldsymbol{s}_i$ to one or more words $\boldsymbol{t}_i$ in another language. Entries were taken from an English-German dictionary (Richter, 2008) a Chinese-English dictionary (Denisowski, 1997), and a Chinese-German dictionary (Hefti, 2005). As with WordNet, the words in entries for English and German were stemmed to improve coverage. An example for German is shown in Figure 2(b).

Algorithmic Connections In addition to hand-curated connections across languages, one could also consider automatic means of mapping across languages, such as using edit distance or local context (Haghighi et al., 2008; Rapp, 1995) or using a lexical translation table obtained from parallel text (Melamed, 1998). While we experimented



(a) GermaNet (b) Dictionary

Figure 2: Two methods for constructing multilingual distributions over words. On the left, paths to the German word "wunsch" in GermaNet are shown. On the right, paths to the English word "room" are shown. Both English and German words are shown; some internal nodes in GermaNet have been omitted for space (represented by dashed lines). Note that different senses are denoted by different internal paths, and that internal paths are distinct from the per-language expression.

with these techniques, constructing appropriate hierarchies from these resources required many arbitrary decisions about cutoffs and which words to include. Thus, we do not consider them in this paper.

4 Experiments

We evaluate MLSLDA on three criteria: how well it can discover consistent topics across languages for matching parallel documents, how well it can discover sentiment-correlated word lists from non-aligned text, and how well it can predict sentiment.

4.1 Matching on Multilingual Topics

We took the 1996 documents from the Europarl corpus (Koehn, 2005) using three bridges: GermaNet, dictionary, and the uninformative flat matching.[4] The model is unaware that the translations of documents in one language are present in the other language. Note that this does not use the supervised framework (as there is no associated response variable for Europarl documents); this experiment is to demonstrate the effectiveness of the multilingual aspect of the model. To test whether the topics learned by the model are consistent across languages, we represent each document using the probability distribution θ_d over topic assignments. Each θ_d is a vector of length K and is a language-independent representation of the document.

For each document in one language, we computed the Hellinger distance between it and all of the documents in the other language and sorted the documents by decreasing distance. The translation of the document is somewhere in that set; the higher the normalized rank (the percentage of documents with a rank lower than the translation of the document), the better the underlying topic model connects languages.

We compare three bridges against what is to our knowledge the only other topic model for unaligned text, Multilingual Topics for Unaligned Text (Boyd-Graber and Blei, 2009).[5]

[4] For English and German documents in all experiments, we removed stop words (Loper and Bird, 2002), stemmed words (Porter and Boulton, 1970), and created a vocabulary of the most frequent 5000 words per language (this vocabulary limit was mostly done to ensure that the dictionary-based bridge was of manageable size). Documents shorter than fifty content words were excluded.

[5] The bipartite matching was initialized with the dictionary weights as specified by the Multilingual Topics for Unaligned Text algorithm. The matching size was limited to 250 and the bipartite matching was only updated on the initial iteration then held fixed. This yielded results comparable to when the matching

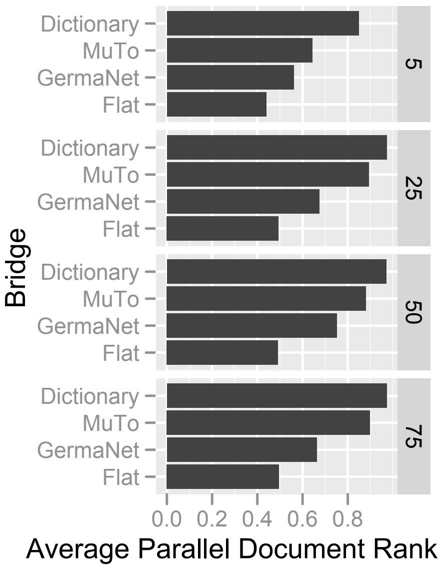


Figure 3: Average rank of paired translation document recovered from the multilingual topic model. Random guessing would yield 0.5; MLSLDA with a dictionary based matching performed best.

Figure 3 shows the results of this experiment. The dictionary-based bridge had the best performance on the task, ranking a large proportion of documents (0.95) below the translated document once enough topics were available. Although GermaNet is richer, its coverage is incomplete; the dictionary structure had a much larger vocabulary and could build a more complete multilingual topics. Using comparable input information, this more flexible model performed better on the matching task than the existing multilingual topic model available for unaligned text. The degenerate flat bridge did no better than the baseline of random guessing, as expected.

4.2 Qualitative Sentiment-Correlated Topics

One of the key tasks in sentiment analysis has been the collection of lists of words that convey sentiment (Wilson, 2008; Riloff et al., 2003). These resources are often created using or in reference to resources like WordNet (Whitelaw et al., 2005; Baccianella and Sebastiani, 2010). MLSLDA provides a method for extracting topical and sentiment-correlated word lists from multilingual corpora. If a WordNet-like resource is used as the bridge, the resulting topics are distributions over synsets, not just over words.

As our demonstration corpus, we used the Amherst Sentiment Corpus (Constant et al., 2009), as it has documents in multiple languages (English, Chinese, and German) with numerical assessments of sentiment (number of stars assigned to the review). We segmented the Chinese text (Tseng et al., 2005) and used a classifier trained on character n-grams to remove English-language documents that were mixed in among the Chinese and German language reviews.

Figure 4 shows extracted topics from German-English and German-Chinese corpora. MLSLDA is able to distinguish sentiment-bearing topics from content bearing topics. For example; in the German-English corpus, “food” and “children” topics are not associated with a consistent sentiment signal, while “religion” is associated with a more negative sentiment. In contrast, in the German-Chinese corpus, the “religion/society” topic is more neutral, and the gender-oriented topic is viewed more negatively. Negative sentiment-bearing topics have reasonable words such as “pages,” “kǒng pà” (Chinese for “I'm afraid that . . . ”) and “tuo” (Chienese for “discard”), and positive sentiment-bearing topics have reasonable words such as “great,” “good,” and “juwel” (German for “jewel”).

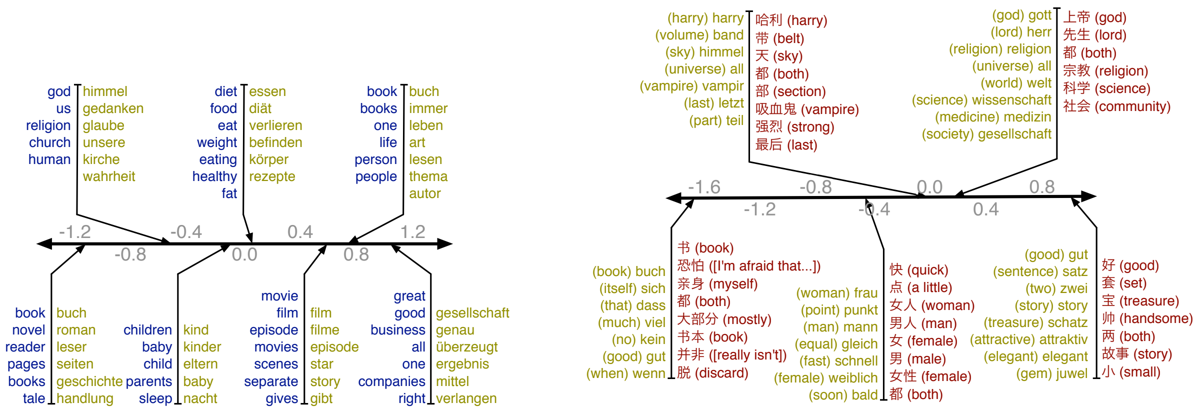
The qualitative topics also betray some of the weaknesses of the model. For example, in one of the negative sentiment topics, the German word “gut” (good) is present. Because topics are distributions over words, they can encode the presence of negations like “kein” (no) and “nicht” (not), but not collocations like “nicht gut.” More elaborate topic models that can model local syntax and collocations (Johnson, 2010) provide options for addressing such problems.

We do not report the results for sentiment prediction for this corpus because the baseline of predicting a positive review is so strong; most algorithms do extremely well by always predicting a positive review, ours included.

4.3 Sentiment Prediction

We gathered 330 film reviews from a German film review site (Vetter et al., 2000) and combined them with a much larger English film review corpus of over

was updated more frequently.



(a) German / English

(b) German / Chinese

Figure 4: Topics, along with associated regression coefficient η from a learned 25-topic model on German-English (left) and German-Chinese (right) documents. Notice that theme-related topics have regression parameter near zero, topics discussing the number of pages have negative regression parameters, topics with "good," "great," "hǎo" (good) and "überzeugt" (convinced) have positive regression parameters. For the German-Chinese corpus, note the presence of "gut" (good) in one of the negative sentiment topics, showing the difficulty of learning collocations.

Train	Test	GermaNet	Dictionary	Flat
DE	DE	73.8	24.8	92.2
EN	DE	7.44	2.68	18.3
EN + DE	DE	**1.17**	**1.46**	**1.39**

Table 1: Mean squared error on a film review corpus. All results are on the same German test data, varying the training data. Over-fitting prevents the model learning on the German data alone; adding English data to the mix allows the model to make better predictions.

5000 film reviews (Pang and Lee, 2005) to create a multilingual film review corpus.[6]

The results for predicting sentiment in German documents with 25 topics are presented in Table 1. On a small monolingual corpus, prediction is very poor. The model over-fits, especially when it has the entire vocabulary to select from. The slightly better performance using GermaNet and a dictionary as topic priors can be viewed as basic feature selection, removing proper names from the vocabulary to prevent over-fitting.

One would expect that prediction improves with a larger training set. For this model, such an improvement is seen even when the training set includes *no* documents in the target language. Note that even the degenerate flat bridge across languages provides useful information. After introducing English data, the model learns to prefer smaller regression parameters (this can be seen as a form of regularization).

Performance is best when a reasonably large corpus is available including some data in the target language. For each bridge, performance improves dramatically, showing that MLSLDA is successfully able to incorporate information learned from both languages to build a single, coherent picture of how sentiment is expressed in both languages. With the GermaNet bridge, performance is better than both the degenerate and dictionary based bridges, showing that the model is sharing information both through the multilingual topics and the regression parameters. Performance on English prediction is comparable to previously published results on this dataset (Blei and McAuliffe, 2007); with enough data, a monolingual model is no longer helped by adding additional multilingual data.

[6] We followed Pang and Lee's method for creating a numerical score between 0 and 1 from a star rating. We then converted that to an integer by multiplying by 100; this was done because initial data preprocessing assumed integer values (although downstream processing did not assume integer values). The German movie review corpus is available at `http://www.umiacs.umd.edu/~jbg/static/downloads_and_media.html`

5 Relationship to Previous Research

The advantages of MLSLDA reside largely in the assumptions that it makes and does not make: documents need not be parallel, sentiment is a normally distributed document-level property, words are exchangeable, and sentiment can be predicted as a regression on a K-dimensional vector.

By not assuming parallel text, this approach can be applied to a broad class of corpora. Other multilingual topic models require parallel text, either at the document (Ni et al., 2009; Mimno et al., 2009) or word-level (Kim and Khudanpur, 2004; Zhao and Xing, 2006). Similarly, other multilingual sentiment approaches also require parallel text, often supplied via automatic translation; after the translated text is available, either monolingual analysis (Denecke, 2008) or co-training is applied (Wan, 2009). In contrast, our approach requires fewer resources for a language: a dictionary (or similar knowledge structure relating words to nodes in a graph) and comparable text, instead of parallel text or a machine translation system.

Rather than viewing one language through the lens of another language, MLSLDA views all languages through the lens of the topics present in a document. This is a modeling decision with pros and cons. It allows a language agnostic decision about sentiment to be made, but it restricts the expressiveness of the model in terms of sentiment in two ways. First, it throws away information important to sentiment analysis like syntactic constructions (Greene and Resnik, 2009) and document structure (McDonald et al., 2007) that may impact the sentiment rating. Second, a single real number is not always sufficient to capture the nuances of sentiment. Less critically, assuming that sentiment is normally distributed is not true of all real-world corpora; review corpora often have a skew toward positive reviews. We standardize responses by the mean and variance of the training data to partially address this issue, but other response distributions are possible, such as generalized linear models (Blei and McAuliffe, 2007) and vector machines (Zhu et al., 2009), which would allow more traditional classification predictions.

Other probabilistic models for sentiment classification view sentiment as a word level feature. Some models use sentiment word lists, either given or learned from a corpus, as a prior to seed topics so that they attract other sentiment bearing words (Mei et al., 2007; Lin and He, 2009). Other approaches view sentiment or perspective as a perturbation of a log-linear topic model (Lin et al., 2008). Such techniques could be combined with the multilingual approach presented here by using distributions over words that not only bridge different languages but also encode additional information. For example, the vocabulary hierarchies could be structured to encourage topics that encourage correlation among similar sentiment-bearing words (e.g. clustering words associated with price, size, etc.). Future work could also more rigorously validate that the multilingual topics discovered by MLSLDA are sentiment-bearing via human judgments.

In contrast, MLSLDA draws on techniques that view sentiment as a regression problem based on the topics used in a document, as in supervised latent Dirichlet allocation (SLDA) (Blei and McAuliffe, 2007) or in finer-grained parts of a document (Titov and McDonald, 2008). Extending these models to multilingual data would be more straightforward.

6 Conclusions

MLSLDA is a "holistic" statistical model for multilingual corpora that does not require parallel text or expensive multilingual resources. It discovers connections across languages that can recover latent structure in parallel corpora, discover sentiment-correlated word lists in multiple languages, and make accurate predictions across languages that improve with more multilingual data, as demonstrated in the context of sentiment analysis.

More generally, MLSLDA provides a formalism that can be used to incorporate the many insights of topic modeling-driven sentiment analysis to multilingual corpora by tying together word distributions across languages. MLSLDA can also contribute to the development of word list-based sentiment systems: the topics discovered by MLSLDA can serve as a first-pass means of sentiment-based word lists for languages that might lack annotated resources.

MLSLDA also can be viewed as a sentiment-informed multilingual word sense disambiguation (WSD) algorithm. When the multilingual bridge is an explicit representation of sense such as WordNet, part of the generative process is an explicit assignment of every word to sense (the path latent variable λ); this is discovered during inference. The dictionary-based technique may be viewed as a disambiguation via a transfer dictionary. How sentiment prediction impacts the implicit WSD is left to future work.

Better capturing local syntax and meaningful collocations would also improve the model's ability to predict sentiment and model multilingual topics, as would providing a better mechanism for representing words not included in our bridges. We intend to develop such models as future work.

7 Acknowledgments

This research was funded in part by the Army Research Laboratory through ARL Cooperative Agreement W911NF-09-2-0072 and by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), through the Army Research Laboratory. All statements of fact, opinion or conclusions contained herein are those of the authors and should not be construed as representing the official views or policies of ARL, IARPA, the ODNI, or the U.S. Government. The authors thank the anonymous reviewers, Jonathan Chang, Christiane Fellbaum, and Lawrence Watts for helpful comments. The authors especially thank Chris Potts for providing help in obtaining and processing reviews.

References

David Andrzejewski, Xiaojin Zhu, and Mark Craven. 2009. Incorporating domain knowledge into topic modeling via Dirichlet forest priors. In *ICML*.

Andrea Esuli Stefano Baccianella and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*.

Carmen Banea, Rada Mihalcea, Janyce Wiebe, and Samer Hassan. 2008. Multilingual subjectivity analysis using machine translation. In *EMNLP*.

David M. Blei and John D. Lafferty. 2005. Correlated topic models. In *NIPS*.

David M. Blei and Jon D. McAuliffe. 2007. Supervised topic models. In *NIPS*. MIT Press.

David M. Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet allocation. *JMLR*, 3:993–1022.

Jordan Boyd-Graber and David M. Blei. 2009. Multilingual topic models for unaligned text. In *UAI*.

Jordan Boyd-Graber, David M. Blei, and Xiaojin Zhu. 2007. A topic model for word sense disambiguation. In *EMNLP*.

Jonathan Chang and David M. Blei. 2009. Relational topic models for document networks. In *AISTATS*.

Shay B. Cohen and Noah A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *NAACL*.

Noah Constant, Christopher Davis, Christopher Potts, and Florian Schwarz. 2009. The pragmatics of expressive content: Evidence from large corpora. *Sprache und Datenverarbeitung*, 33(1–2).

Kerstin Denecke. 2008. Using SentiWordNet for multilingual sentiment analysis. In *ICDEW 2008*.

Paul Denisowski. 1997. CEDICT. http://www.mdbg.net/chindict/.

Jean Diebolt and Eddie H.S. Ip, 1996. *Markov Chain Monte Carlo in Practice*, chapter Stochastic EM: method and application. Chapman and Hall, London.

Alexander Geyken. 2007. The DWDS corpus: A reference corpus for the German language of the 20th century. In *Idioms and Collocations: Corpus-based Linguistic, Lexicographic Studies*. Continuum Press.

Stephan Greene and Philip Resnik. 2009. More than words: Syntactic packaging and implicit sentiment. In *NAACL*.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *PNAS*, 101(Suppl 1):5228–5235.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *ACL*, Columbus, Ohio.

Jan Hefti. 2005. HanDeDict. http://chdw.de.

Hitoshi Isahara, Fransis Bond, Kiyotaka Uchimoto, Masao Utiyama, and Kyoko Kanzaki. 2008. Development of the Japanese WordNet. In *LREC*.

Mark Johnson. 2010. PCFGs, topic models, adaptor grammars and learning topical collocations and the structure of proper names. In *ACL*.

Woosung Kim and Sanjeev Khudanpur. 2004. Lexical triggers and latent semantic analysis for cross-lingual language model adaptation. *TALIP*, 3(2):94–112.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*. http://www.statmt.org/europarl/.

Claudia Kunze and Lothar Lemnitzer. 2002. Standardizing WordNets in a web-compliant format: The case of GermaNet. In *Workshop on Wordnets Structures and Standardisation*.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *CIKM*.

Wei-Hao Lin, Eric Xing, and Alexander Hauptmann. 2008. A joint topic and perspective model for ideological discourse. In *ECML PKDD*.

Edward Loper and Steven Bird. 2002. NLTK: the natural language toolkit. In *Tools and methodologies for teaching*. ACL.

Ryan McDonald, Kerry Hannan, Tyler Neylon, Mike Wells, and Jeff Reynar. 2007. Structured models for fine-to-coarse sentiment analysis. In *ACL*.

Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. 2007. Topic sentiment mixture: modeling facets and opinions in weblogs. In *WWW*.

Ilya Dan Melamed. 1998. *Empirical methods for exploiting parallel texts*. Ph.D. thesis, University of Pennsylvania.

George A. Miller. 1990. Nouns in WordNet: A lexical inheritance system. *International Journal of Lexicography*, 3(4):245–264.

David Mimno, Hanna Wallach, Jason Naradowsky, David Smith, and Andrew McCallum. 2009. Polylingual topic models. In *EMNLP*.

Xiaochuan Ni, Jian-Tao Sun, Jian Hu, and Zheng Chen. 2009. Mining multilingual topics from Wikipedia. In *WWW*.

Noam Ordan and Shuly Wintner. 2007. Hebrew WordNet: a test case of aligning lexical databases across languages. *International Journal of Translation*, 19(1):39–58.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*.

Bo Pang and Lillian Lee. 2008. *Opinion Mining and Sentiment Analysis*. Now Publishers Inc.

Martin Porter and Richard Boulton. 1970. Snowball stemmer. http://snowball.tartarus.org/credits.php.

Matthew Purver, Konrad Körding, Thomas L. Griffiths, and Joshua Tenenbaum. 2006. Unsupervised topic modelling for multi-party spoken discourse. In *ACL*.

Reinhard Rapp. 1995. Identifying word translations in non-parallel texts. In *ACL*, pages 320–322.

Philip Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI*, pages 448–453.

Frank Richter. 2008. Dictionary nice grep. http://www-user.tu-chemnitz.de/ fri/ding/.

Ellen Riloff, Janyce Wiebe, and Theresa Wilson. 2003. Learning subjective nouns using extraction pattern bootstrapping. In *NAACL*.

Michal Rosen-Zvi, Thomas L. Griffiths, Mark Steyvers, and Padhraic Smyth. 2004. The author-topic model for authors and documents. In *UAI*.

Benoît Sagot and Darja Fišer. 2008. Building a Free French WordNet from Multilingual Resources. In *OntoLex*.

Ivan Titov and Ryan McDonald. 2008. A joint model of text and aspect ratings for sentiment summarization. In *ACL*, pages 308–316.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter. In *SIGHAN Workshop on Chinese Language Processing*.

University of Oxford. 2006. British National Corpus. http://www.natcorp.ox.ac.uk/. http://www.natcorp.ox.ac.uk/.

Tobias Vetter, Manfred Sauer, and Philipp Wallutat. 2000. Filmrezension.de: Online-magazin für filmkritik. http://www.filmrezension.de.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *ACL*.

Chong Wang, David Blei, and Li Fei-Fei. 2009. Simultaneous image classification and annotation. In *CVPR*.

Xing Wei and Bruce Croft. 2006. LDA-based document models for ad-hoc retrieval. In *SIGIR*.

Casey Whitelaw, Navendu Garg, and Shlomo Argamon. 2005. Using appraisal groups for sentiment analysis. In *CIKM*.

Theresa Ann Wilson. 2008. *Fine-grained Subjectivity and Sentiment Analysis: Recognizing the Intensity, Polarity, and Attitudes of Private States*. Ph.D. thesis, University of Pittsburgh.

Bing Zhao and Eric P. Xing. 2006. BiTAM: Bilingual topic admixture models for word alignment. In *ACL*.

Jun Zhu, Amr Ahmed, and Eric P. Xing. 2009. Medlda: maximum margin supervised topic models for regression and classification. In *ICML*.

Jointly Modeling Aspects and Opinions with a MaxEnt-LDA Hybrid

Wayne Xin Zhao[†], **Jing Jiang**[‡], **Hongfei Yan**[†], **Xiaoming Li**[†]
[†]School of Electronics Engineering and Computer Science, Peking University, China
[‡]School of Information Systems, Singapore Management University, Singapore
{zhaoxin,yhf}@net.pku.edu.cn, jingjiang@smu.edu.cn, lxm@pku.edu.cn

Abstract

Discovering and summarizing opinions from online reviews is an important and challenging task. A commonly-adopted framework generates structured review summaries with aspects and opinions. Recently topic models have been used to identify meaningful review aspects, but existing topic models do not identify aspect-specific opinion words. In this paper, we propose a MaxEnt-LDA hybrid model to jointly discover both aspects and aspect-specific opinion words. We show that with a relatively small amount of training data, our model can effectively identify aspect and opinion words simultaneously. We also demonstrate the domain adaptability of our model.

1 Introduction

With the dramatic growth of opinionated user-generated content, consumers often turn to online product reviews to seek advice while companies see reviews as a valuable source of consumer feedback. How to automatically understand, extract and summarize the opinions expressed in online reviews has therefore become an important research topic and gained much attention in recent years (Pang and Lee, 2008). A wide spectrum of tasks have been studied under review mining, ranging from coarse-grained document-level polarity classification (Pang et al., 2002) to fine-grained extraction of opinion expressions and their targets (Wu et al., 2009). In particular, a general framework of summarizing reviews of a certain product is to first identify different aspects (a.k.a. features) of the given product and then extract specific opinion expressions for each aspect. For example, aspects of a restaurant may include *food*, *staff*, *ambience* and *price*, and opinion expressions for *staff* may include *friendly*, *rude*, etc. Because of the practicality of this structured summary format, it has been adopted in several previous studies (Hu and Liu, 2004; Popescu and Etzioni, 2005; Brody and Elhadad, 2010) as well as some commercial systems, e.g. the "scorecard" feature at Bing shopping[1].

Different approaches have been proposed to identify aspect words and phrases from reviews. Previous methods using frequent itemset mining (Hu and Liu, 2004) or supervised learning (Jin and Ho, 2009; Jin et al., 2009; Wu et al., 2009) have the limitation that they do not group semantically related aspect expressions together. Supervised learning also suffers from its heavy dependence on training data. In contrast, unsupervised, knowledge-lean topic modeling approach has been shown to be effective in automatically identifying aspects and their representative words (Titov and McDonald, 2008; Brody and Elhadad, 2010). For example, words such as *waiter*, *waitress*, *staff* and *service* are grouped into one aspect.

We follow this promising direction and extend existing topic models to jointly identify both aspect and opinion words, especially aspect-specific opinion words. Current topic models for opinion mining, which we will review in detail in Section 2, still lack this ability. But separating aspect and opinion words can be very useful. Aspect-specific opinion words can be used to construct a domain-dependent senti-

[1] http://www.bing.com/shopping

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 56–65,
MIT, Massachusetts, USA, 9-11 October 2010.

ment lexicon and applied to tasks such as sentiment classification. They can also provide more informative descriptions of the product or service being reviewed. For example, using more specific opinion words such as *cozy* and *romantic* to describe the *ambience* aspect in a review summary is more meaningful than using generic words such as *nice* and *great*. To the best of our knowledge, Brody and Elhadad (2010) are the first to study aspect-specific opinion words, but their opinion word detection is performed outside of topic modeling, and they only consider adjectives as possible opinion words.

In this paper, we propose a new topic modeling approach that can automatically separate aspect and opinion words. A novelty of this model is the integration of a discriminative maximum entropy (MaxEnt) component with the standard generative component. The MaxEnt component allows us to leverage arbitrary features such as POS tags to help separate aspect and opinion words. Because the supervision relies mostly on non-lexical features, although our model is no longer fully unsupervised, the number of training sentences needed is relatively small. Moreover, training data can also come from a different domain and yet still remain effective, making our model highly domain adaptive. Empirical evaluation on large review data sets shows that our model can effectively identify both aspects and aspect-specific opinion words with a small amount of training data.

2 Related Work

Pioneered by the work of Hu and Liu (2004), review summarization has been an important research topic. There are usually two major tasks involved, namely, aspect or feature identification and opinion extraction. Hu and Liu (2004) applied frequent itemset mining to identify product features without supervision, and considered adjectives collocated with feature words as opinion words. Jin and Ho (2009), Jin et al. (2009) and Wu et al. (2009) used supervised learning that requires hand-labeled training sentences to identify both aspects and opinions. A common limitation of these methods is that they do not group semantically related aspect expressions together. Furthermore, supervised learning usually requires a large amount of training data in order to perform well and is not easily domain adaptable.

Topic modeling provides an unsupervised and knowledge-lean approach to opinion mining. Titov and McDonald (2008) show that global topic models such as LDA (Blei et al., 2003) may not be suitable for detecting rateable aspects. They propose multi-grain topic models for discovering local rateable aspects. However, they do not explicitly separate aspect and opinion words. Lin and He (2009) propose a joint topic-sentiment model, but topic words and sentiment words are still not explicitly separated. Mei et al. (2007) propose to separate topic and sentiment words using a positive sentiment model and a negative sentiment model, but both models capture general opinion words only. In contrast, we model *aspect-specific* opinion words as well as general opinion words.

Recently Brody and Elhadad (2010) propose to detect aspect-specific opinion words in an unsupervised manner. They take a two-step approach by first detecting aspect words using topic models and then identifying aspect-specific opinion words using polarity propagation. They only consider adjectives as opinion words, which may potentially miss opinion words with other POS tags. We try to jointly capture both aspect and opinion words within topic models, and we allow non-adjective opinion words.

Another line of related work is about how to incorporate useful features into topic models (Zhu and Xing, 2010; Mimno and McCallum, 2008). Our MaxEnt-LDA hybrid bears similarity to these recent models but ours is designed for opinion mining.

3 Model Description

Our model is an extension of LDA (Blei et al., 2003) but captures both aspect words and opinion words. To model the aspect words, we use a modified version of the multi-grain topic models from (Titov and McDonald, 2008). Our model is simpler and yet still produces meaningful aspects. Specifically, we assume that there are T aspects in a given collection of reviews from the same domain, and each review document contains a mixture of aspects. We further assume that each sentence (instead of each word as in standard LDA) is assigned to a single aspect, which is often true based on our observation.

To understand how we model the opinion words, let us first look at two example review sentences from the restaurant domain:

The food was tasty.
The waiter was quite friendly.

We can see that there is a strong association of *tasty* with *food* and similarly of *friendly* with *waiter*. While both *tasty* and *friendly* are specific to the restaurant domain, they are each associated with only a single aspect, namely *food* and *staff*, respectively. Besides these aspect-specific opinion words, we also see general opinion words such as *great* in the sentence "*The food was great!*" These general opinion words are shared across aspects, as opposed to aspect-specific opinion words which are used most commonly with their corresponding aspects. We therefore introduce a general opinion model and T aspect-specific opinion models to capture these different opinion words.

3.1 Generative Process

We now describe the generative process of the model. First, we draw several multinomial word distributions from a symmetric Dirichlet prior with parameter β: a background model $\phi^{\mathcal{B}}$, a general aspect model $\phi^{\mathcal{A},g}$, a general opinion model $\phi^{\mathcal{O},g}$, T aspect models $\{\phi^{\mathcal{A},t}\}_{t=1}^{T}$ and T aspect-specific opinion models $\{\phi^{\mathcal{O},t}\}_{t=1}^{T}$. All these are multinomial distributions over the vocabulary, which we assume has V words. Then for each review document d, we draw a topic distribution $\theta^d \sim \text{Dir}(\alpha)$ as in standard LDA. For each sentence s in document d, we draw an aspect assignment $z_{d,s} \sim \text{Multi}(\theta^d)$.

Now for each word in sentence s of document d, we have several choices: The word may describe the specific aspect (e.g. *waiter* for the *staff* aspect), or a general aspect (e.g. *restaurant*), or an opinion either specific to the aspect (e.g. *friendly*) or generic (e.g. *great*), or a commonly used background word (e.g. *know*). To distinguish between these choices, we introduce two indicator variable, $y_{d,s,n}$ and $u_{d,s,n}$, for the nth word $w_{d,s,n}$. We draw $y_{d,s,n}$ from a multinomial distribution over $\{0, 1, 2\}$, parameterized by $\pi^{d,s,n}$. $y_{d,s,n}$ determines whether $w_{d,s,n}$ is a background word, aspect word or opinion word. We will discuss how to set $\pi^{d,s,n}$ in Section 3.2. We draw $u_{d,s,n}$ from a Bernoulli distribution over $\{0, 1\}$ parameterized by p, which in turn is drawn from a symmetric Beta(γ). $u_{d,s,n}$ determines whether $w_{d,s,n}$ is general or aspect-specific. We then draw $w_{d,s,n}$ as follows:

$$w_{d,s,n} \sim \begin{cases} \text{Multi}(\phi^{\mathcal{B}}) & \text{if } y_{d,s,n} = 0 \\ \text{Multi}(\phi^{\mathcal{A},z_{d,s}}) & \text{if } y_{d,s,n} = 1, u_{d,s,n} = 0 \\ \text{Multi}(\phi^{\mathcal{A},g}) & \text{if } y_{d,s,n} = 1, u_{d,s,n} = 1 \\ \text{Multi}(\phi^{\mathcal{O},z_{d,s}}) & \text{if } y_{d,s,n} = 2, u_{d,s,n} = 0 \\ \text{Multi}(\phi^{\mathcal{O},g}) & \text{if } y_{d,s,n} = 2, u_{d,s,n} = 1 \end{cases}.$$

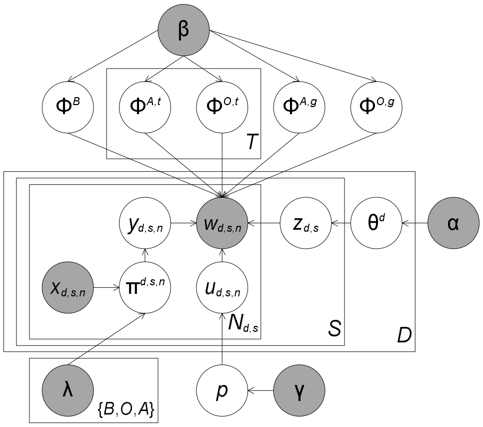


Figure 1: The plate notation of our model.

Figure 1 shows our model using the plate notation.

3.2 Setting π with a Maximum Entropy Model

A simple way to set $\pi^{d,s,n}$ is to draw it from a symmetric Dirichlet prior. However, as suggested in (Mei et al., 2007; Lin and He, 2009), fully unsupervised topic models are unable to identify opinion words well. An important observation we make is that aspect words and opinion words usually play different syntactic roles in a sentence. Aspect words tend to be nouns while opinion words tend to be adjectives. Their contexts in sentences can also be different. But we do not want to use strict rules to separate aspect and opinion words because there are also exceptions. E.g. verbs such as *recommend* can also be opinion words.

In order to use information such as POS tags to help discriminate between aspect and opinion words, we propose a novel idea as follows: We set $\pi^{d,s,n}$ using a maximum entropy (MaxEnt) model applied to a feature vector $\boldsymbol{x}_{d,s,n}$ associated with $w_{d,s,n}$. $\boldsymbol{x}_{d,s,n}$ can encode any arbitrary features we think may be discriminative, e.g. previous, current and next POS tags. Formally, we have

$$p(y_{d,s,n} = l|\boldsymbol{x}_{d,s,n}) = \pi_l^{d,s,n} = \frac{\exp\left(\lambda_l \cdot \boldsymbol{x}_{d,s,n}\right)}{\sum_{l'=0}^{2} \exp\left(\lambda_{l'} \cdot \boldsymbol{x}_{d,s,n}\right)},$$

where $\{\lambda_l\}_{l=0}^{2}$ denote the MaxEnt model weights and can be learned from a set of training sentences with labeled background, aspect and opinion words. This MaxEnt-LDA hybrid model is partially inspired by (Mimno and McCallum, 2008).

As for the features included in $\boldsymbol{x}$, currently we use two types of simple features: (1) lexical features which include the previous, the current and the next words $\{\mathrm{w}_{i-1}, \mathrm{w}_i, \mathrm{w}_{i+1}\}$, and (2) POS tag features which include the previous, the current and the next POS tags $\{\mathrm{POS}_{i-1}, \mathrm{POS}_i, \mathrm{POS}_{i+1}\}$.

3.3 Inference

We use Gibbs sampling to perform model inference. Due to the space limit, we leave out the derivation details and only show the sampling formulas. Note that the MaxEnt component is trained first independently of the Gibbs sampling procedure, that is, in Gibbs sampling, we assume that the λ parameters are fixed.

We use $\boldsymbol{w}$ to denote all the words we observe in the collection, $\boldsymbol{x}$ to denote all the feature vectors for these words, and $\boldsymbol{y}$, $\boldsymbol{z}$ and $\boldsymbol{u}$ to denote all the hidden variables. First, given the assignment of all other hidden variables, to sample a value for $z_{d,s}$, we use the following formula:

$$
\begin{aligned}
& P(z_{d,s} = t | \boldsymbol{z}_{\neg(d,s)}, \boldsymbol{y}, \boldsymbol{u}, \boldsymbol{w}, \boldsymbol{x}) \propto \frac{c^{d}_{(t)} + \alpha}{c^{d}_{(\cdot)} + T\alpha} \\
& \times \left(\frac{\Gamma\left(c^{\mathcal{A},t}_{(\cdot)} + V\beta\right)}{\Gamma\left(c^{\mathcal{A},t}_{(\cdot)} + n^{\mathcal{A},t}_{(\cdot)} + V\beta\right)} \cdot \prod_{v=1}^{V} \frac{\Gamma\left(c^{\mathcal{A},t}_{(v)} + n^{\mathcal{A},t}_{(v)} + \beta\right)}{\Gamma\left(c^{\mathcal{A},t}_{(v)} + \beta\right)} \right) \\
& \times \left(\frac{\Gamma\left(c^{\mathcal{O},t}_{(\cdot)} + V\beta\right)}{\Gamma\left(c^{\mathcal{O},t}_{(\cdot)} + n^{\mathcal{O},t}_{(\cdot)} + V\beta\right)} \cdot \prod_{v=1}^{V} \frac{\Gamma\left(c^{\mathcal{O},t}_{(v)} + n^{\mathcal{O},t}_{(v)} + \beta\right)}{\Gamma\left(c^{\mathcal{O},t}_{(v)} + \beta\right)} \right).
\end{aligned}
$$

Here $c^{d}_{(t)}$ is the number of sentences assigned to aspect t in document d, and $c^{d}_{(\cdot)}$ is the number of sentences in document d. $c^{\mathcal{A},t}_{(v)}$ is the number of times word v is assigned as an *aspect* word to aspect t, and $c^{\mathcal{O},t}_{(v)}$ is the number of times word v is assigned as an *opinion* word to aspect t. $c^{\mathcal{A},t}_{(\cdot)}$ is the total number of times any word is assigned as an aspect word to aspect t, and $c^{\mathcal{O},t}_{(\cdot)}$ is the total number of times any word is assigned as an opinion word to aspect t. All these counts represented by a c variable exclude sentence s of document d. $n^{\mathcal{A},t}_{(v)}$ is the number of times word v is assigned as an aspect word to aspect t in sentence s of document d, and similarly, $n^{\mathcal{O},t}_{(v)}$ is the number of times word v is assigned as an opinion word to aspect t in sentence s of document d.

Then, to jointly sample values for $y_{d,s,n}$ and $u_{d,s,n}$, we have

$$
\begin{aligned}
& P(y_{d,s,n} = 0 | \boldsymbol{z}, \boldsymbol{y}_{\neg(d,s,n)}, \boldsymbol{u}_{\neg(d,s,n)}, \boldsymbol{w}, \boldsymbol{x}) \\
\propto\; & \frac{\exp(\lambda_0 \cdot \boldsymbol{x}_{d,s,n})}{\sum_{l'} \exp(\lambda_{l'} \cdot \boldsymbol{x}_{d,s,n})} \cdot \frac{c^{\mathcal{B}}_{(w_{d,s,n})} + \beta}{c^{\mathcal{B}}_{(\cdot)} + V\beta}, \\
& P(y_{d,s,n} = l, u_{d,s,n} = b | \boldsymbol{z}, \boldsymbol{y}_{\neg(d,s,n)}, \boldsymbol{u}_{\neg(d,s,n)}, \boldsymbol{w}, \boldsymbol{x}) \\
\propto\; & \frac{\exp(\lambda_l \cdot \boldsymbol{x}_{d,s,n})}{\sum_{l'} \exp(\lambda_{l'} \cdot \boldsymbol{x}_{d,s,n})} \cdot g(w_{d,s,n}, z_{d,s}, l, b),
\end{aligned}
$$

where the function $g(v, t, l, b)$ $(1 \le v \le V, 1 \le t \le T, l \in \{1, 2\}, b \in \{0, 1\})$ is defined as follows:

$$
g(v,t,l,b) = \begin{cases}
\frac{c^{\mathcal{A},t}_{(v)} + \beta}{c^{\mathcal{A},t}_{(\cdot)} + V\beta} \cdot \frac{c_{(0)} + \gamma}{c_{(\cdot)} + 2\gamma} & \text{if } l = 1, b = 0 \\
\frac{c^{\mathcal{O},t}_{(v)} + \beta}{c^{\mathcal{O},t}_{(\cdot)} + V\beta} \cdot \frac{c_{(0)} + \gamma}{c_{(\cdot)} + 2\gamma} & \text{if } l = 2, b = 0 \\
\frac{c^{\mathcal{A},g}_{(v)} + \beta}{c^{\mathcal{A},g}_{(\cdot)} + V\beta} \cdot \frac{c_{(1)} + \gamma}{c_{(\cdot)} + 2\gamma} & \text{if } l = 1, b = 1 \\
\frac{c^{\mathcal{O},g}_{(v)} + \beta}{c^{\mathcal{O},g}_{(\cdot)} + V\beta} \cdot \frac{c_{(1)} + \gamma}{c_{(\cdot)} + 2\gamma} & \text{if } l = 2, b = 1.
\end{cases}.
$$

Here the various c variables denote various counts excluding the nth word in sentence s of document d. Due to space limit, we do not give full explanation here.

4 Experiment Setup

To evaluate our MaxEnt-LDA hybrid model for jointly modeling aspect and opinion words, we used a restaurant review data set previously used in (Ganu et al., 2009; Brody and Elhadad, 2010) and a hotel review data set previously used in (Baccianella et al., 2009). We removed stop words and used the Stanford POS Tagger[2] to tag the two data sets. Only reviews that have no more than 50 sentences were used. We also kept another version of the data which includes the stop words for the purpose of extracting the contextual features included in $\boldsymbol{x}$. Some details of the data sets are given in Table 1.

For our hybrid model, we ran 500 iterations of Gibbs sampling. Following (Griffiths and Steyvers, 2004), we fixed the Dirichlet priors as follows: $\alpha =$

[2] http://nlp.stanford.edu/software/tagger.shtml

data set	restaurant	hotel
#tokens	1,644,923	1,097,739
#docs	52,574	14,443

Table 1: Some statistics of the data sets.

data set	#sentences	#tokens
restaurant	46	634
cell phone	125	4414
DVD player	180	3024

Table 2: Some statistics of the labeled training data.

$50/T$, $\beta = 0.1$ and $\gamma = 0.5$. We also experimented with other settings of these priors and did not notice any major difference. For MaxEnt training, we tried three labeled data sets: one that was taken from the restaurant data set and manually annotated by us[3], and two from the annotated data set used in (Wu et al., 2009). Note that the latter two were used for testing domain adaptation in Section 6.3. Some details of the training sets are shown in Table 2.

In our preliminary experiments, we also tried two variations of our MaxEnt-LDA hybrid model. (1) The first is a fully unsupervised model where we used a uniform Dirichlet prior for π. We found that this unsupervised model could not separate aspect and opinion words well. (2) The second is a bootstrapping version of the MaxEnt-LDA model where we used the predicted values of y as pseudo labels and re-trained the MaxEnt model iteratively. We found that this bootstrapping procedure did not boost the overall performance much and even hurt the performance a little in some cases. Due to the space limit we do not report these experiments here.

5 Evaluation

In this section we report the evaluation of our model. We refer to our MaxEnt-LDA hybrid model as *ME-LDA*. We also implemented a local version of the standard LDA method where each sentence is treated as a document. This is the model used in (Brody and Elhadad, 2010) to identify aspects, and we refer to this model as *LocLDA*.

Food	Staff	Order Taking	Ambience
chocolate	service	wait	room
dessert	food	waiter	dining
cake	staff	wait	tables
cream	**excellent**	order	bar
ice	**friendly**	minutes	place
desserts	**attentive**	seated	decor
coffee	**extremely**	waitress	scene
tea	waiters	reservation	space
bread	**slow**	asked	area
cheese	**outstanding**	told	table

Table 4: Sample aspects of the restaurant domain using LocLDA. Note that the words in bold are opinion words which are mixed with aspect words.

5.1 Qualitative Evaluation

For each of the two data sets, we show four sample aspects identified by ME-LDA in Table 3 and Table 5. Because the hotel domain is somehow similar to the restaurant domain, we used the labeled training data from the restaurant domain also for the hotel data set. From the tables we can see that generally aspect words are quite coherent and meaningful, and opinion words correspond to aspects very well. For comparison, we also applied LocLDA to the restaurant data set and present the aspects in Table 4. We can see that ME-LDA and LocLDA give similar aspect words. The major difference between these two models is that ME-LDA can sperate aspect words and opinion words, which can be very useful. ME-LDA is also able to separate general opinion words from aspect-specific ones, giving more informative opinion expressions for each aspect.

5.2 Evaluation of Aspects Identification

We also quantitatively evaluated the quality of the automatically identified aspects. Ganu et al. (2009) provide a set of annotated sentences from the restaurant data set, in which each sentence has been assigned one or more labels from a gold standard label set $\mathcal{S}$ = {Staff, Food, Ambience, Price, Anecdote, Misc}. To evaluate the quality of our aspect identification, we chose from the gold standard labels three major aspects, namely *Staff*, *Food* and *Ambience*. We did not choose the other aspects because (1) *Price* is often mixed with other aspects such as *Food*, and (2) *Anecdote* and *Misc* do not show clear

[3] We randomly selected 46 sentences for manual annotation.

Food		Staff		Order Taking		Ambience		General
Aspect	Opinion	Aspect	Opinion	Aspect	Opinion	Aspect	Opinion	Opinion
chocolate	good	service	friendly	table	seated	room	small	good
dessert	best	staff	attentive	minutes	asked	dining	nice	well
cake	great	food	great	wait	told	tables	beautiful	nice
cream	delicious	wait	nice	waiter	waited	bar	romantic	great
ice	sweet	waiter	good	reservation	waiting	place	cozy	better
desserts	hot	place	excellent	order	long	decor	great	small
coffee	amazing	waiters	helpful	time	arrived	scene	open	bad
tea	fresh	restaurant	rude	hour	rude	space	warm	worth
bread	tasted	waitress	extremely	manager	sat	area	feel	definitely
cheese	excellent	waitstaff	slow	people	finally	table	comfortable	special

Table 3: Sample aspects and opinion words of the restaurant domain using ME-LDA.

Service		Room Condition		Ambience		Meal		General
Aspect	Opinion	Aspect	Opinion	Aspect	Opinion	Aspect	Opinion	Opinion
staff	helpful	room	shower	room	quiet	breakfast	good	great
desk	friendly	bathroom	small	floor	open	coffee	fresh	good
hotel	front	bed	clean	hotel	small	fruit	continental	nice
english	polite	air	comfortable	noise	noisy	buffet	included	well
reception	courteous	tv	hot	street	nice	eggs	hot	excellent
help	pleasant	conditioning	large	view	top	pastries	cold	best
service	asked	water	nice	night	lovely	cheese	nice	small
concierge	good	rooms	safe	breakfast	hear	room	great	lovely
room	excellent	beds	double	room	overlooking	tea	delicious	better
restaurant	rude	bath	well	terrace	beautiful	cereal	adequate	fine

Table 5: Sample aspects and opinion words of the hotel domain using ME-LDA.

patterns in either word usage or writing styles, making it even hard for humans to identify them. Brody and Elhadad (2010) also only used these three aspects for quantitative evaluation. To avoid ambiguity, we used only the single-labeled sentences for evaluation. About 83% of the labeled sentences have a single label, which confirms our observation that a sentence usually belongs to a single aspect.

Aspect	Method	Precision	Recall	F-1
Staff	LocLDA	0.804	0.585	0.677
	ME-LDA	0.779	0.540	0.638
Food	LocLDA	0.898	0.648	0.753
	ME-LDA	0.874	0.787	0.828
Ambience	LocLDA	0.603	0.677	0.638
	ME-LDA	0.773	0.558	0.648

Table 6: Results of aspects identification on restaurant.

We first ran ME-LDA and LocLDA each to get an inferred aspect set $\mathcal{T}$. Following (Brody and Elhadad, 2010), we set the number of aspects to 14 in both models. We then manually mapped each inferred aspect to one of the six gold standard aspects, i.e., we created a mapping function $f(t) : \mathcal{T} \rightarrow \mathcal{S}$. For sentence s of document d, we first assign it to an inferred aspect as follows:

$$t^* = \arg\max_{t \in \mathcal{T}} \sum_{n=1}^{N_{d,s}} \log P(w_{d,s,n}|t).$$

We then assign the gold standard aspect $f(t^*)$ to this sentence. We then calculated the F-1 score of the three aspects: *Staff*, *Food* and *Ambience*. The results are shown in Table 6. Generally ME-LDA has given competitive results compared with LocLDA. For *Food* and *Ambience* ME-LDA outperformed LocLDA, while for *Staff* ME-LDA is a little worse than LocLDA. Note that ME-LDA is not designed to compete with LocLDA for aspect identification.

5.3 Evaluation of Opinion Identification

Since the major advantage of ME-LDA is its ability to separate aspect and opinion words, we further quantitatively evaluated the quality of the aspect-specific opinion words identified by ME-LDA. Brody and Elhadad (2010) has constructed a gold standard set of aspect-specific opinion words for the restaurant data set. In this gold standard set, they manually judged eight out of the 14 automatically inferred aspects they had: $\mathcal{J}$ = {Ambiance, Staff, Food-Main Dishes, Atmosphere-Physical, Food-Baked Goods, Food-General, Drinks, Service}. Each word is assigned a polarity score ranging from -2.0 to 2.0 in each aspect. We used their gold standard words whose polarity scores are not equal to zero. Because their gold standard only includes adjectives, we also manually added more opinion words into the gold standard set. To do so, we took the top 20 opinion words returned by our method and two baseline methods, pooled them together, and manually judged them. We use precision at n (P@n), a commonly used metric in information retrieval, for evaluation. Because top words are more important in opinion models, we set n to 5, 10 and 20. For both ME-LDA and BL-1 below, we again manually mapped each automatically inferred aspect to one of the gold standard aspects.

Since LocLDA does not identify aspect-specific opinion words, we consider the following two baseline methods that can identify aspect-specific opinion words:

BL-1: In this baseline, we start with all adjectives as candidate opinion words, and use mutual information (MI) to rank these candidates. Specifically, given an aspect t, we rank the candidate words according to the following scoring function:

$$\text{Score}_{\text{BL-1}}(w,t) = \sum_{v \in \mathcal{V}_t} p(w,v) \log \frac{p(w,v)}{p(w)p(v)},$$

where $\mathcal{V}_t$ is the set of the top-100 frequent aspect words from $\phi^{A,t}$.

BL-2: In this baseline, we first use LocLDA to learn a topic distribution for each sentence. We then assign a sentence to the aspect with the largest probability and hence get sentence clusters. We manually map these clusters to the eight gold standard aspects. Finally, for each aspect we rank adjectives by their frequencies in the aspect and treat these as aspect-specific opinion words.

Method	P@5	P@10	P@20
ME-LDA	$0.825^{*,\diamond}$	0.700^{*}	0.569^{*}
BL-1	0.400	0.450	0.469
BL-2	0.725	0.650	0.563

Table 7: Average P@n of aspect-specific opinion words on restaurant. * and $\diamond$ indicate that the improvement hypothesis is accepted at confidence level 0.9 respectively for BL-1 and BL-2.

The basic results in terms of the average precision at n over the eight aspects are shown in Table 7. We can see that ME-LDA outperformed the two baselines consistently. Especially, for P@5, ME-LDA gave more than 100% relative improvement over BL-1. The absolute value of 0.825 for P@5 also indicates that top opinion words discovered by our model are indeed meaningful.

5.4 Evaluation of the Association between Opinion Words and Aspects

The evaluation in the previous section shows that our model returns good opinion words for each aspect. It does not, however, directly judge how *aspect-specific* those opinion words are. This is because the gold standard created by (Brody and Elhadad, 2010) also includes general opinion words. E.g. *friendly* and *good* may both be judged to be opinion words for the *staff* aspect, but the former is more specific than the latter. We suspect that BL-2 has comparable performance with ME-LDA for this reason. So we further evaluated the association between opinion words and aspects by directly looking at how easy it is to infer the corresponding aspect by only looking at an aspect-specific opinion word. We selected four aspects for evaluation: *Ambiance*, *Staff*, *Food-Main Dishes* and *Atmosphere-Physical* . We chose these four aspects because they are quite different from each other and thus manual judgments on these four aspects can be more objective. For each aspect, similar to the pooling strategy in IR, we pooled the top 20 opinion words identified by BL-1, BL-2 and ME-LDA. We then asked two human assessors to assign an association score to each of these words as follows: If the word is closely associated with an aspect, a score of 2 is given; if it is marginally as-

Metrics	Dataset	BL-2	ME-LDA
nDCG@5	Restaurant	0.647	0.764
	Hotel	0.782	0.820
nDCG@10	Restaurant	0.781	0.897
	Hotel	0.722	0.789

Table 8: Average nDCG performance of BL-2 and ME-LDA. Because only four aspects were used for evaluation, we did not perform statistical significance test. We found that in all cases ME-LDA outperformed BL-2 for either all aspects or three out of four aspects.

sociated with an aspect, a score of 1 is given; otherwise, 0 is given. We calculated the Kappa statistics of agreement, and we got a quite high Kappa value of 0.8375 and 0.7875 respectively for the restaurant data set and the hotel data set. Then for each word in an aspect, we took the average of the scores of the two assessors. We used an nDCG-like metric to compare the performance of our model and of BL-2. The metric is defined as follows:

$$\text{nDCG@}k(t, \mathcal{M}) = \frac{\sum_{i=1}^{k} \frac{\text{Score}(\mathcal{M}_{t,i})}{\log_2(i+1)}}{\text{iDCG@}k(t)},$$

where $\mathcal{M}_{t,i}$ is the ith aspect-specific opinion word inferred by method $\mathcal{M}$ for aspect t, Score($\mathcal{M}_{t,i}$) is the association score of this word, and iDCG@$k(t)$ is the score of the ideal DCG measure at k for aspect t, that is, the maximum DCG score assuming an ideal ranking. We chose $k = 5$ and $k = 10$. The average nDCG over the four aspects are presented in Table 8. We can see that ME-LDA outperformed BL-2 quite a lot for the restaurant data set, which conforms to our hypothesis that ME-LDA generates aspect-specific opinion words of stronger association with aspects. For the hotel data set, ME-LDA outperformed a little. This may be due to the fact that we used the restaurant training data for the hotel data set.

6 Further Analysis of MaxEnt

In this section, we perform some further evaluation and analysis of the MaxEnt component in our model.

6.1 Feature Selection

Previous studies have shown that simple POS features and lexical features can be very effective for discovering aspect words and opinion words (Hu and Liu, 2004; Jin et al., 2009; Wu et al., 2009; Brody and Elhadad, 2010). for POS features, since we observe that aspect words tend to be nouns while opinion words tend to be adjectives but sometimes also verbs or other part-of-speeches, we can expect that POS features should be quite useful. As for lexical features, words from a sentiment lexicon can also be helpful in discovering opinion words.

Methods	Average F-1
LocLDA	0.690
ME-LDA + $\mathcal{A}$	0.631
ME-LDA + $\mathcal{B}$	0.695
ME-LDA + $\mathcal{C}$	0.705

Table 9: Comparison of the average F-1 using different feature sets for aspect identification on restaurant.

However, lexical features are more diverse so presumably we need more training data in order to detect useful lexical features. Lexical features are also more domain-dependent. On the other hand, we hypothesize that POS features are more effective when the amount of training data is small and/or the training data comes from a different domain. We therefore compare the following three sets of features:

- $\mathcal{A}$: w_{i-1}, w_i, w_{i+1}
- $\mathcal{B}$: POS_{i-1}, POS_i, POS_{i+1}
- $\mathcal{C}$: $\mathcal{A} + \mathcal{B}$

We show the comparison of the performance in Table 9 using the average F-1 score defined in Section 5.2 for aspect identification, and in Table 10 using the average P@n measure defined in Section 5.3 for opinion identification. We can see that Set $\mathcal{B}$ plays the most important part, which conforms to our hypothesis that POS features are very important in opinion mining. In addition, we can see that Set $\mathcal{C}$ performs a bit better than Set $\mathcal{B}$, which indicates that some lexical features (e.g., general opinion words) may also be helpful. Note that here the training data is from the same domain as the test data, and therefore lexical features are likely to be useful.

6.2 Examine the Size of Labeled Data

As we have seen, POS features play the major role in discriminating between aspect and opinion words. Because there are much fewer POS features than word features, we expect that we do not need many

Methods	P@5	P@10	P@20
BL-2	0.725	0.650	0.563
ME-LDA + $\mathcal{A}$	0.150	0.200	0.231
ME-LDA + $\mathcal{B}$	0.775	0.688	0.569
ME-LDA + $\mathcal{C}$	0.825	0.700	0.569

Table 10: Comparison of the average P@n using different feature sets for opinion identification on restaurant.

Method	F-1
LocalLDA	0.690
ME-LDA + 10	0.629
ME-LDA + 20	0.692
ME-LDA + 30	0.691
ME-LDA + 40	0.726
ME-LDA + 46	0.705

Table 11: Average F-1 with differen sizes of training data on restaurant.

labeled sentences to learn the POS-based patterns. We now examine the sensitivity of the performance with respect to the amount of labeled data. We generated four smaller training data sets with 10, 20, 30 and 40 sentences each from the whole training data set we have, which consists of 46 labeled sentences. The results are shown in Table 11 and Table 12. We can see that generally the performance stays above BL when the number of training sentences is 20 or more. This indicates that our model needs only a relatively small number of high-quality training sentences to achieve good results.

6.3 Domain Adaption

Since we find that the MaxEnt supervision relies more on POS features than lexical features, we also hypothesize that if the training sentences come from a different domain the performance can still remain relatively high. To test this hypothesis, we tried two quite different training data sets, one from the *cell phone* domain and the other from the *DVD player* domain, both used in (Wu et al., 2009).

Method	P@5	P@10	P@20
BL-2	0.725	0.650	0.563
ME-LDA + 10	0.700	0.563	0.488
ME-LDA + 20	0.875	0.650	0.600
ME-LDA + 30	0.825	0.700	0.569
ME-LDA + 40	0.825	0.688	0.581
ME-LDA + 46	0.825	0.700	0.569

Table 12: Average P@n of aspect-specific opinion words with differen sizes of training data on restaurant.

Method	Average F-1
restaurant + $\mathcal{B}$	0.695
restaurant + $\mathcal{C}$	0.705
cell phone + $\mathcal{B}$	0.662
cell phone + $\mathcal{C}$	0.629
DVD player + $\mathcal{B}$	0.686
DVD player + $\mathcal{C}$	0.635

Table 13: Average F-1 performance for domain adaption on restaurant.

Method	P@5	P@10	P@20
restaurant + $\mathcal{B}$	0.775	0.688	0.569
restaurant + $\mathcal{C}$	0.825	0.700	0.569
cell phone + $\mathcal{B}$	0.775	0.675	0.588
cell phone + $\mathcal{C}$	0.750	0.688	0.594
DVD player + $\mathcal{B}$	0.775	0.713	0.575
DVD player + $\mathcal{C}$	0.825	0.663	0.588

Table 14: Average P@n of aspect-specific opinion words for domain adaption on restaurant.

We consider two feature sets defined in Section 6.1 for domain adaption, namely $\mathcal{B}$ and $\mathcal{C}$. The results are shown in Table 13 and Table 14.

For aspect identification, using out-of-domain training data performed worse than using in-domain training data, but the absolute performance is still decent. And interestingly, we can see that using $\mathcal{B}$ is better than using $\mathcal{C}$, indicating that lexical features may hurt the performance in the cross-domain setting. It suggests that lexical features are not easily adaptable across domains for aspect identification.

For opinion identification, we can see that there is no clear difference between using out-of-domain training data and using in-domain training data, which may indicate that our opinion identification component is robust in domain adaption. Also, we cannot easily tell whether $\mathcal{B}$ has advantage over $\mathcal{C}$ for opinion identification. One possible reason may be that those general opinion words are useful across domains, so lexical features may still be useful for domain adaption.

7 Conclusions

In this paper, we presented a topic modeling approach that can jointly identify aspect and opinion words, using a MaxEnt-LDA hybrid. We showed that by incorporating a supervised, discriminative maximum entropy model into an unsupervised, generative topic model, we could leverage syntactic features to help separate aspect and opinion words. We evaluated our model on two large review data sets from the restaurant and the hotel domains. We found that our model was competitive in identifying meaningful aspects compared with previous models. Most importantly, our model was able to identify meaningful opinion words strongly associated with different aspects. We also demonstrated that the model could perform well with a relatively small amount of training data or with training data from a different domain.

Our model provides a principled way to jointly model both aspects and opinions. One of the future directions we plan to explore is to use this model to help sentence-level extraction of specific opinions and their targets, which previously was only tackled in a fully supervised manner. Another direction is to extend the model to support polarity classification.

ACKNOWLEDGMENT

The authors Xin Zhao, Hongfei Yan and Xiaoming Li are partially supported by NSFC under the grant No. 70903008 and 60933004, CNGI grant No. 2008-122, 863 Program No. 2009AA01Z143, and the Open Fund of the State Key Laboratory of Software Development Environment under Grant No. SKLSDE-2010KF-03, Beihang University.

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2009. Multi-facet rating of product reviews. In *Proceedings of the 31st ECIR*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3.

Samuel Brody and Noemie Elhadad. 2010. An unsupervised aspect-sentiment model for online reviews. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Gayatree Ganu, Noemie Elhadad, and Amelie Marian. 2009. Beyond the stars: Improving rating predictions using review text content. In *Proceedings of the 12th International Workshop on the Web and Databases*.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Wei Jin and Hung Hay Ho. 2009. A novel lexicalized HMM-based learning framework for web opinion mining. In *Proceedings of the 26th International Conference on Machine Learning*.

Wei Jin, Hung Hay Ho, and Rohini K. Srihari. 2009. OpinionMiner: A novel machine learning system for web opinion mining and extraction. In *Proceedings of the 15th ACM SIGKDD*.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceeding of the Eighteenth ACM Conference on Information and Knowledge Management*.

Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. 2007. Topic sentiment mixture: Modeling facets and opinions in weblogs. In *Proceedings of the 16th International Conference on World Wide Web*.

David Mimno and Andrew McCallum. 2008. Topic models conditioned on arbitrary features with dirichlet-multinomial regression. In *Conference on Uncertainty in Artificial Intelligence*.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2).

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

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of the HLT-EMNLP*.

Ivan Titov and Ryan McDonald. 2008. Modeling online reviews with multi-grain topic models. In *Proceeding of the 17th International Conference on World Wide Web*.

Yuanbin Wu, Qi Zhang, Xuangjing Huang, and Lide Wu. 2009. Phrase dependency parsing for opinion mining. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*.

Jun Zhu and Eric P. Xing. 2010. Conditional topic random fields. In *Proceedings of the 27th International Conference on Machine Learning*.

Summarizing Contrastive Viewpoints in Opinionated Text

Michael J. Paul[*]
University of Illinois
Urbana, IL 61801, USA
mjpaul2@illinois.edu

ChengXiang Zhai
University of Illinois
Urbana, IL 61801, USA
czhai@cs.uiuc.edu

Roxana Girju
University of Illinois
Urbana, IL 61801, USA
girju@illinois.edu

Abstract

This paper presents a two-stage approach to summarizing multiple contrastive viewpoints in opinionated text. In the first stage, we use an unsupervised probabilistic approach to model and extract multiple viewpoints in text. We experiment with a variety of lexical and syntactic features, yielding significant performance gains over bag-of-words feature sets. In the second stage, we introduce *Comparative LexRank*, a novel random walk formulation to score sentences and pairs of sentences from opposite viewpoints based on both their representativeness of the collection as well as their contrastiveness with each other. Experimental results show that the proposed approach can generate informative summaries of viewpoints in opinionated text.

1 Introduction

The amount of opinionated text available online has been growing rapidly, increasing the need for systems that can summarize opinions expressed in such text so that a user can easily digest them. In this paper, we study how to summarize opinionated text in a such a way that highlights contrast between multiple viewpoints, which is a little-studied task.

Usually, online opinionated text is generated by multiple people, and thus often contains multiple viewpoints about an issue or topic. A viewpoint/perspective refers to "a mental position from which things are viewed" (cf. WordNet). An opinion is usually expressed in association with a particular viewpoint, even though the viewpoint is usually not explicitly given; for example, a blogger that is in favor of a policy would likely look at the positive aspects of the policy (i.e., positive viewpoint), while someone against the policy would likely emphasize the negative aspects (i.e., negative viewpoint). Moreover, in an opinionated text with diverse opinions, the multiple viewpoints taken by opinion holders are often "contrastive", leading to opposite polarities. Indeed, such contrast in opinions may be a main driving force behind many online discussions.

Futhermore, opinions regarding news events and other short-term issues may quickly emerge and disappear. Such opinions may reflect many different types of viewpoints which cannot be modeled by current systems. For this reason, we believe that a viewpoint summarization system would benefit from the ability to extract unlabeled viewpoints without supervision. Even if such clustering has inaccuracies, it could still be a useful starting point for human editors to select representative excerpts.

Thus, given a set of opinionated documents about a topic, we aim at automatically extracting and summarizing the multiple contrastive viewpoints implicitly expressed in the opinionated text to facilitate digestion and comparison of different viewpoints. Specifically, we will generate two types of multi-view summaries: macro multi-view summary and micro multi-view summary. A macro multi-view summary would contain multiple sets of sentences, each representing a different viewpoint; these different sets of sentences can be compared to understand the difference of multiple viewpoints at the "macro level." A micro multi-view summary would contain a set of pairs of contrastive sentences (each pair

[*]Now at Johns Hopkins University (mpaul@cs.jhu.edu).

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 66–76,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

consists of two sentences representing two different viewpoints), making it easy to understand the difference between two viewpoints at the "micro level."

Although opinion summarization has been extensively studied (e.g., (Liu et al., 2005; Hu and Liu, 2004; Hu and Liu, 2006; Zhuang et al., 2006)), existing work has not attempted to generate our envisioned contrastive macro and micro multi-view summaries in an unsupervised way, which is the goal of our work. For example, Hu and Liu (2006) rank sentences based on their dominant sentiment according to the polarity of adjectives occuring near a product feature in a sentence. A contradiction occurs when two sentences are highly unlikely to be simultaneously true (cf. (Marneffe et al., 2008)). Although little work has been done on contradiction detection, there are a few notable approaches (Harabagiu et al., 2006; Marneffe et al., 2008; Kim and Zhai, 2009).

The closest work to ours is perhaps that of Lerman and McDonald (2009) who present an approach to contrastive summarization. They add an objective to their summarization model such that the summary model for one set of text is different from the model for the other set. The idea is to highlight the key differences between the sets, however this is a different type of contrast than the one we study here – our goal is instead to make the summaries *similar* to each other, to contrast how the same information is conveyed through different viewpoints.

In this paper, we propose a two-stage approach to solving this novel summarization problem, which will be explained in the following two sections.

2 Modeling Viewpoints

The first challenge to be solved in order to generate a contrastive summary of multiple viewpoints is to model and extract these viewpoints which are hidden in text. In this paper we propose to solve this challenge by employing the Topic-Aspect Model (TAM) (Paul and Girju, 2010), which is an extension of the Latent Dirichlet Allocation (LDA) model (Blei et al., 2003) for jointly modeling topics and viewpoints in text. While most existing work on such topic models (including TAM) has taken a topic model as a generative model for word tokens in text, we propose to take TAM as a generative model for more complex linguistic features extracted from text. These are more discriminative than single word tokens and can improve the accuracy of extracting multiple viewpoints as we will show in the experimental results' section. Below we first give a brief introduction to TAM and then present the proposed set of features.

2.1 Topic-Aspect Model (TAM)

LDA-style probabilistic topic models of document content (Blei et al., 2003) have been shown to offer state-of-the-art summarization quality. Such models also provide a framework for adding additional structure to a summarization model (Haghighi and Vanderwende, 2009). In our case, we want to add more structure to a model to incorporate the notion of viewpoint/perspective into our summaries.

When it comes to extracting viewpoints, recent research suggests that it may be beneficial to model both topics and perspectives, as sentiment may be expressed differently depending on the issue involved (Brody and Elhadad, 2010; Paul and Girju, 2010). For example, let's consider a set of product reviews for a home theater system. Content topics in this data might include things like sound quality, usability, etc., while the viewpoints might be the positive and negative sentiments. A word like *speakers*, for instance depends on the sound topic but not a viewpoint, while *good* would be an example of a word that depends on a viewpoint but not any particular topic. A word like *loud* would depend on both (since it would be considered positive sentiment only in the context of the sound quality topic), while a word like *think* depends on neither.

We make use of a recent model, the Topic-Aspect Model (Paul and Girju, 2010), which can model such behavior with or without supervision. Under this model, a document has a mixture over topics as well as a mixture over viewpoints. The two mixtures are drawn independently of each other, and thus can be thought of as two separate clustering dimensions. A word is associated with variables denoting its topic and viewpoint assignments, as well as two binary variables to denote if the word depends on the topic and if the word depends on the viewpoint. A word may depend on the topic, the viewpoint, both, or neither, as in the above example.

The generative process for a document d under this model can be briefly described as follows. For each word in a document:

1. Sample a topic z from $P(z|d)$ and a viewpoint v from $P(v|d)$.
2. Sample a "level" $\ell \in \{0,1\}$ from $P(\ell|d)$. This determines if the word will depend on the topic (topical level) or not (background level).
3. Sample a "route" $r \in \{0,1\}$ from $P(r|\ell,z)$. This determines if the word will depend on the viewpoint.
4. Sample a word w from $P(w|z,v,r,\ell)$.

The probabilities are multinomial/binomial distributions with Dirichlet/Beta priors, and thus this model falls under the standard LDA framework. The number of topics and number of viewpoints are parameters that must be specified. Inference can be done with Gibbs sampling (Paul and Girju, 2010).

TAM naturally gives us a very rich output to use in a viewpoint summarization application. If we are doing unsupervised viewpoint extraction, we can use the output of the model to compute $P(v|sentence)$ which could be used to generate summaries that contain only excerpts that strongly highlight one viewpoint over another. Similarly, we could use the learned topic mixtures to generate topic-specific summaries. Futhermore, the variables r and ℓ tell us if a word is dependent on the viewpoint and topic, and we could use this information to focus on sentences that contain informative content words. Note that without supervision, TAM's clustering is based only on co-occurrences and the patterns it captures may or may not correspond with the viewpoints we wish to extract. Nonetheless, we show in this research that it can indeed find meaningful viewpoints with reasonable accuracy on certain data sets. Although we do not explore this in this paper, additional information about the viewpoints could be added to TAM by defining priors on the distributions to further improve the accuracy of viewpoint discovery.

2.2 Features

Previous work with TAM used only bag of words features, which may not be the best features for capturing viewpoints. For example, "Israel attacked Palestine" and "Palestine attacked Israel" are identical excerpts in an exchangable bag of words representation, yet one is more likely to come from the perspective of a Palestinian and the other from an Israeli. In this subsection, we will propose a variety of feature sets. We evaluate the utility of these features to the task of modeling viewpoints by measuring the accuracy of unsupervised clustering.

2.2.1 Words

We have experimented with simple bag of words features as baseline approaches, both with and without removing stop words, and found that the accuracy of clustering by viewpoint is better when retaining all words. This supports the observation that common function words may have important psychological properties (Chung and Pennebaker, 2007). Thus, we do not do any stop word removal for any of our other feature sets. We find that we get better results by stemming the words, so we apply Porter's stemmer to all of our features described.

2.2.2 Dependency Relations

It has been shown that using syntactic information can improve the accuracy of sentiment models (Joshi and Rosé, 2009). Thus, instead of representing documents as a bag of words, we will experiment with using features returned by a dependency parser. For this, we used the Stanford parser[1], which returns dependency tuples of the form $\texttt{rel}(a,b)$ where $\texttt{rel}$ is some dependency relation and a and b are tokens of a sentence. We can use these specific tuples as features, referred here as the *full-tuple* representation.

One problem with this representation is that we are using very specific information and it is harder for learning algorithms to find patterns due to the lack of redundancy. One solution is to generalize these features and rewrite a tuple $\texttt{rel}(a,b)$ as two tuples: $\texttt{rel}(a,*)$ and $\texttt{rel}(*,b)$ (Greene and Resnik, 2009; Joshi and Rosé, 2009). We will refer to this as the *split-tuple* representation.

2.2.3 Negation

If a word w_i appears in the head of a $\texttt{neg}$ relation, then we would like this to be reflected in other dependency tuples in which w_i occurs. For a tuple $\texttt{rel}(w_i,w_j)$, if either w_i or w_j is negated, then we simply rewrite it as $\neg\texttt{rel}(w_i,w_j)$.

An alternative would be to rewrite the individual word w_i as $\neg w_i$. However in our experiments this representation produced worse accuracies, perhaps because this produces less redundancy.

[1] http://nlp.stanford.edu/software/

2.2.4 Polarity

We also hypothesize that lexical polarity information may improve our model. If we are using the *full-tuple* representation, then a tuple becomes more general by replacing the specific word with a $+$ or $-$. In the case that both words are polarity words, we use two tuples, replacing only one word at a time rather than replacing both words with their polarity signs. To determine the polarity of a word, we simply use the Subjectivity Clues lexicon (Wilson et al., 2005) and as polarity values, *positive* (+), *negative* (-), and *neutral* (*). Under our *split-tuple* representation, this becomes more specific by replacing the $*$ with the polarity sign. For example, the tuple $\texttt{amod}(idea, good)$ would be represented as $\texttt{amod}(idea, +)$ and $\texttt{amod}(*, good)$. We collapse negated features to flip the polarity sign such that $\neg\texttt{rel}(a, +)$ becomes $\texttt{rel}(a, -)$.

2.2.5 Generalized Relations

We also experimented with backing off the relations themselves. Since the Stanford dependencies can be organized in a hierarchy[2], we will represent the relations at more generalized levels in the hierarchy. For example, both a direct object and an indirect object are a type of object. For a relation $\texttt{rel}$, we define $\texttt{R}_{\texttt{rel}}$ as the relation above $\texttt{rel}$ in the hierarchy – for example, $\texttt{R}_{\texttt{dobj}} = \texttt{obj}$. We make an exception for neg which has its own important properties that we wish to retain, so we let $\texttt{R}_{\texttt{neg}} = \texttt{neg}$. Thus, when using these features, we rewrite $\texttt{rel}(a, b)$ as $\texttt{R}_{\texttt{rel}}(a, b)$.

3 Multi-Viewpoint Summarization

As a computation problem, extractive multi-viewpoint summarization would take as input a set of candidate excerpts[3] $X = \{x_1, x_2, ..., x_{|X|}\}$ with k viewpoints and generate two types of multi-view contrastive summaries: 1) A macro contrastive summary S_{macro} consists of k disjoint sets of excerpts, $X_1, X_2, ..., X_k \subset X$ with each X_i containing representative sentences of the i-th view (i.e., $S_{macro} = (X_1, ..., X_k)$). The number of excerpts in each X_i can be empirically set based on application needs. 2) A micro contrastive summary S_{micro} consists of a set of excerpt pairs, each containing two excerpts from two different viewpoints, i.e., $S_{micro} = \{(s_1, t_1), ..., (s_n, t_n)\}$ where $s_i \in X$ and $t_i \in X$ are two comparable excerpts representing two different viewpoints. n is the length of the summary, which can be set empirically based on application needs. Note that both macro and micro summaries can reveal contrast between different viewpoints, though at different granularity levels.

To generate macro and micro summaries based on the probabilistic assignment of excerpts to viewpoints given by TAM, we propose a novel extension to the LexRank algorithm (Erkan and Radev, 2004), a graph-based method for scoring representative excerpts to be used in a summary. Our key idea is to modify the definition of the jumping probability in the random walk model so that it would favor excerpts that represent a viewpoint well and encourage jumping to an excerpt comparable with the current one but from a different viewpoint. As a result, the stationary distribution of the random walk model would capture representative contrastive excerpts and allow us to generate both macro and micro contrastive summaries within a unified framework. We now describe this novel summarization algorithm (called Comparative LexRank) in detail.

3.1 Comparative LexRank

LexRank is a PageRank-like algorithm (Page et al., 1998), where we define a random walk model on top of a graph that has sentences to be summarized as nodes and edges placed between two sentences that are similar to each other. We can then score all the sentences based on the expected probability of a random walker visiting each sentence. We use the short-hand $P(x_j|x_i)$ to denote the probability of being at node x_j at a time t given that the walker was at x_i at time $t-1$. The jumping probability from node x_i to node x_j is given by:

$$P(x_j|x_i) = \frac{sim(x_i, x_j)}{\sum_{j' \in X} sim(x_i, x_{j'})} \quad (1)$$

where sim is a content similarity function defined on two sentence/excerpt nodes.

Our extension is mainly to modify this jumping probability in two ways so as to favor visiting contrastive representative opinions from multiple view-

[2] The complete hierarchy can be found in the Stanford dependencies manual (Marneffe and Manning, 2008).

[3] An "excerpt" refers to the smallest unit of text that will make up our summary such as a sentence.

points. The first modification is to make it favor jumping to a good representative excerpt x of any viewpoint v (i.e., with high probability $p(v|x)$ according to the TAM model). The second modification is to further favor jumping between two excerpts that can potentially form a good contrastive pair for use in generating a micro contrastive summary.

Specifically, under our model, the random walker first decides whether to jump to a sentence of the same viewpoint or to a sentence of a different viewpoint. We define this decision as a binary variable $z \in \{0, 1\}$. Intuitively, if we can force the random walker to move back and forth between viewpoints, then the final scores will favor sentences that are similar across both viewpoints.

We define two different modified similarity functions for the two possible values of z. The first one, sim_0 (corresponding to $z = 0$) scales the similarity by the likelihood that the two x's represent the same viewpoint, and the second one, sim_1 (for $z = 1$) scales the similarity by the likelihood that the x's come from different viewpoints.

$$\begin{aligned} sim_0(x_i, x_j) &= sim(x_i, x_j) \sum_{m=1}^{k} P(v = m|x_i)P(v = m|x_j) \\ sim_1(x_i, x_j) &= sim(x_i, x_j) \times \\ &\sum_{m_1, m_2 \in [1,k], m_1 \neq m_2} P(v = m_1|x_i)P(v = m_2|x_j) \end{aligned}$$

where $P(v|x)$ denotes the probability that the excerpt x belongs to the viewpoint v, and in general, can be obtained through any multi-viewpoint model. A special case of this is when the labels for viewpoints are known, in which case $P(v|x) = 1$ for the correct label and 0 for the others.

In our experiments, $P(v|x)$ comes from the output of TAM, and we define $sim(x_i, x_j)$ as the cosine between the vectors x_i and x_j, although again any similarity function could be used. The conditional transition probability from x_i to x_j given z is then:

$$P(x_j|x_i, z) = \frac{sim_z(x_i, x_j)}{\sum_{j' \in X} sim_z(x_i, x_{j'})} \quad (2)$$

Using λ to denote $P(z = 0)$ and marginalizing across z, we have the transition probability:

$$P(x_j|x_i) = \lambda P(x_j|x_i, z = 0) + (1 - \lambda)P(x_j|x_i, z = 1)$$

The stationary distribution of the random walk gives us a scoring of the excerpts to be used in our summary. It is also possible to score *pairs* of excerpts that contrast each other. We define the score for a pair (x_i, x_j) as the probability of being at x_i and transitioning to x_j or vice versa, where x_i and x_j are of opposite viewpoints. Specifically:

$$P(x_i)P(x_j|x_i, z = 1) + P(x_j)P(x_i|x_j, z = 1) \quad (3)$$

3.2 Summary Generation

The final summary should be a set of excerpts that have a high relevance score according to our scoring algorithm, but are not redundant among each other. Many techniques could be used to accomplish this (Carbonell and Goldstein, 1998; McDonald, 2007), but we use a simple greedy approach: at each step of the summary generation algorithm, we add the excerpt with the highest relevance score as long as the excerpt's redundancy score – the cosine similarity between the candidate and the current summary – is under some threshold δ. This is repeated until the summary reaches a user-supplied length limit.

Macro contrastive summarization: A macro-level summary consists of independent summaries for each viewpoint, which we generate by first using the random walk stationary distribution across all of the data to rank the excerpts. We then separate the top-ranked excerpts into two disjoint sets according to their viewpoint based on whichever gives a greater value of $P(v|x)$, and finally remove redundancy and produce the summary according to our method described above. We refer to this as *macro contrastive summarization*, because the summaries will contrast each other in that they have related content, but the excerpts in the summaries are not explicitly aligned with each other.

Micro contrastive summarization: A candidate excerpt for a micro-level summary will consist of a pair (x_i, x_j) with the pairwise relevance score defined in Equation 3. We can then rank these pairs and remove redundancy. It is possible that both x_i and x_j in a high-scoring pair may belong to the same viewpoint; such a case would be filtered out since we are mainly interested in including contrastive pairs in our summary. We refer to this as *micro contrastive summarization*, because the summaries will allow us to see contrast at the level of individual excerpts from different viewpoints.

4 Experiments and Evaluation

4.1 Experimental Setup

Evaluation of multi-view summarization is challenging as there is no existing data set we can use. We leverage the resources on the Web and created two data sets in the domain of political opinion.

Our first dataset is a set of 948 verbatim responses to a Gallup® phone survey about the 2010 U.S. healthcare bill (Jones, 2010), conducted March 4-7, 2010. Responses in this set tend to be short and often incomplete or otherwise ill-formed and informal sentences. Respondants indicate if they are 'for' or 'against' the bill, and there is a roughly even mix of the two viewpoints (45% for and 48% against).

We also use the Bitterlemons corpus, a collection of 594 editorials about the Israel-Palestine conflict. This dataset is fully described in (Lin et al., 2006) and has been used in other perspective modeling literature (Lin et al., 2008; Greene and Resnik, 2009). The style of this data differs substantially from the healthcare data in that documents in this set tend to be long and verbose articles with well-formed sentences. It again contains a fairly even mixture of two different perspectives: 312 articles from Israeli authors and 282 articles from Palestinian authors.

Moreover, for the healthcare data set, manually extracted opinion polls are available on the Web, which we further leverage to construct gold standard summaries to evaluate our method quantitatively. The data and test sets are available at `http://apfel.ai.uiuc.edu/resources.html`.

4.2 Stage One: Modeling Viewpoints

The main research question we want to answer in modeling viewpoints is whether richer feature sets would lead to better accuracy than word features. We used our various feature sets as input to TAM and measured the accuracy of clustering documents by viewpoint. This evaluation serves both to measure how accurately this type of clustering can be done, as well as to measure which types of features are important for modeling viewpoints.

We found that the clustering accuracy is improved if we measure the accuracy of only the subset of documents such that $P(v|doc)$ is greater than some threshold (we used 0.8). Thus, the accuraries presented in this section are measured using this confidence threshold. We will use this approach for the summarization task as well, as it ensures we are only summarizing documents where we have high confidence about their viewpoint membership.

There are several parameters to set for TAM. Since our focus is on comparing linguistic features with word features, we simply set these parameters to some reasonable values: We used Dirichlet pseudo-counts of 80.0 for $P(\ell = 0)$, 20.0 for $P(\ell = 1)$, uniform pseudo-counts of 5.0 for $P(x)$, 0.1 for the topic and aspect mixtures, and 0.01 for the word distributions. We tell the model to use 2 viewpoints as well as 5 topics for the healthcare corpus and 8 topics for the Bitterlemons corpus.

There is high variance in the accuracies depending on how the Gibbs samplers were initialized. We thus repeated the experiments many times to obtain relatively confident measures – 200 times for the healthcare set and 50 times for the Bitterlemons set, with 2000 iterations each time. A natural way to select a model is to choose the model that gives the highest likelihood to its input. To evaluate how well this selection strategy would work, we measured the correlation between accuracy and likelihood.

The results are shown in Table 1. We can make several observations. (1) In all cases, the proposed linguistic features yield higher accuracy than the word features, supporting our hypothesis that for viewpoint modeling, applying TAM to these features improves performance over using simple word features. Since virtually all existing work on topic models assumes word tokens as data to be modeled, our results suggest that it would be interesting to explore applying generative topic models to complex features for other tasks as well. This may be because by adding additional complex features to the observed data, we artificially inflate the data likelihood to emphasize modeling co-occurrences of such features, which effectively biases the model to capture a certain perspective of co-occurrences.

(2) The increase is substantially greater for the Bitterlemons corpus, which may be due to the fact that the parsing accuracy is likely better because the language is formal. The *split-tuple* representation is very significantly better for the healthcare corpus, but it is not clear which is better for the Bitterlemons corpus. It is also not clear how the generalized relations affect the performance.

	Healthcare Corpus					*Bitterlemons Corpus*				
Feature Set	**Mean**	**Med**	**Max**	**MaxLL**	**Corr**	**Mean**	**Med**	**Max**	**MaxLL**	**Corr**
bag of words	61.12 +/- 0.76%	61.01	72.17	52.92	0.187	68.22 +/- 3.31%	69.26	88.27	84.94	0.39
- no stopwords	60.58 +/- 0.79%	60.50	72.18	62.58	0.154	61.29 +/- 3.05%	57.69	91.34	82.91	0.33
full-tuples	62.42 +/- 0.88%	62.47	74.04	63.37	0.201	80.89 +/- 3.45%	85.40	94.07	92.10	0.34
+ negation	63.67 +/- 0.81%	64.54	74.07	69.25	0.338	80.60 +/- 3.88%	88.07	95.61	91.32	0.66
+ neg. + polarity	63.16 +/- 0.94%	64.46	74.05	67.8	0.455	82.53 +/- 3.55%	86.64	94.44	91.16	0.31
gen. full-tuples	63.80 +/- 0.73%	64.35	73.29	71.70	0.254	76.62 +/- 4.09%	84.56	94.53	84.56	0.25
split-tuples	68.32 +/- 0.90%	70.74	77.80	76.57	0.646	77.14 +/- 3.64%	81.29	92.99	88.13	0.30
+ negation	68.00 +/- 0.91%	69.11	79.73	76.14	0.187	83.53 +/- 3.05%	87.71	95.00	95.00	0.12
+ neg. + polarity	65.11 +/- 1.05%	65.35	78.59	67.22	0.159	81.24 +/- 3.37%	83.44	95.03	88.55	0.08
gen. split-tuples	69.31 +/- 0.83%	70.69	77.90	73.90	0.653	76.69 +/- 4.36%	83.78	93.60	91.67	0.09

Table 1: The clustering accuracy with TAM using a variety of feature sets. These results were averaged over 200 randomly-initialized Gibbs sampling procedures for the healthcare set, and 50 procedures for the Bitterlemons set. The 95% confidence interval using a standard t-test is also given. *Max* refers to the maximum accuracy obtained over the 200 or 50 instances. *MaxLL* refers to the clustering accuracy using the model that yielded the highest corpus log-likelihood as defined by TAM. *Corr* refers to the Pearson correlation coefficient between accuracy and log-likelihood.

(3) It appears that adding polarity helps the *full-tuple* features (by making them more general) but hurts the *split-tuple* features (by making them more specific). Negation significantly improves the *full-tuple* features in the Bitterlemons corpus, but it is not clear if it helps in the other cases. It should be noted that capturing negation and polarity is a very complex and difficult task, and it is not expected that our simple approaches will accurately capture these properties. Nonetheless, it seems that these simple features may help in certain cases.

4.3 Stage Two: Summarizing Viewpoints

For the second stage (i.e., the Comparative LexRank algorithm), we mainly want to evaluate the quality of the generated contrastive multi-viewpoint summary and study the effectiveness of our extension to the standard LexRank. Below we present extensive evaluation of our summarization method on the healthcare data. We do not have an evaluation set with which to compute quantitative metrics on the Bitterlemons corpus, so we will instead perform a simple qualitative evaluation in the last subsection.

4.3.1 Gold Standard Summaries

The responses to the Gallup healthcare poll are described in an article[4] which gives a table of the main responses found in the data along with their prominence in the data. In a way, this represents an expert human-generated summary of our database, and we will use this as a gold standard macro contrastive summary against which the representativeness of a multi-viewpoint contrastive summary can be evaluated. The reasons given in this table will be used verbatim as our reference set, excluding the other/no-reason/no-opinion reasons. A sample of this table is shown in Table 2.

We also want to develop a reference set for micro contrastive summaries, where we are mainly interested in evaluating contrastiveness. To do this, we asked 3 annotators to identify contrastive pairs in the “main reasons” table described above. Each pair must contain one reason from the ‘for’ side and one reason from the ’against’ side, though we do not require a one-to-one alignment; that is, multiple pairs may contain the same reason. We take the set of pairs that were identified as being contrastive by at least 2 annotators to be our gold set of contrastive pairs. Because these pairs come from the gold summary, they are still representative of the collection as a whole, rather than fine-grained contrasts.

The macro reference set contains 9 ‘for’ reasons and 15 ‘against’ reasons. The micro reference set contains 13 annotator-identified pairs composed of 9 unique ‘for’ reasons and 8 unique ‘against’ reasons.

4.3.2 Baseline Approaches

Graph-based algorithms: The standard LexRank algorithm can also be used to score pairs of sentences according to Equation 3. We will thus compare our new LexRank extension to the unmodified form of this algorithm. When $\lambda = 1$, the random walk model only transitions to sentences within the same viewpoint, and thus in this case our modified algorithm produces the same ranking as the unmodified LexRank. This will be our first baseline.

[4] http://www.gallup.com/poll/126521/Favor-Oppose-Obama-Healthcare-Plan.aspx

For		*Against*	
People need health insurance/Too many uninsured	29%	Will raise costs of insurance/Make it less affordable	20%
System is broken/Needs to be fixed	18%	Does not address real problems	19%
Costs are out of control/Would help control costs	12%	Need more information/clarity on how system would work	8%
Moral responsibility to provide/Obligation/Fair	12%	Against big government/Too much government involvement	8%

Table 2: Some of the top reasons given along with their prominence in the healthcare data, as analyzed by Gallup. This is a sample of what will serve as our gold set. The highlighted cells show an example of a contrastive pair identified by our annotators.

Model-based algorithms: We will also compare against the approach of Lerman and McDonald (2009) who introduce their contrastiveness objective into a model-based summarization algorithm. The basic form of this algorithm is to select a set of sentences S_m to minimize the KL-divergence between the models of the summary S_m and the entire collection X_m for a viewpoint m. The objective function is: $-\sum_{m=1}^{k} KL(L(S_m)||L(X_m))$ where L is an arbitrary language model. We define $L(A)$ simply as the unigram distribution over words in the collection A, a method also evaluated by Haghighi and Vanderwende (2009). This is the fairest comparison to our LexRank experiments, where sentences are also represented as unigrams. (We do not do any modeling with TAM in our quantitative evaluation.)

Lerman and McDonald introduce an additional term to maximize the KL-divergence between the summary of one viewpoint and the collection of the opposite viewpoint, so that each viewpoint's summary is dissimilar to the other viewpoints. We borrow this idea but instead do the opposite so that the viewpoints' summaries are more (rather than less) similar to each other. This contrastive version of our model-based baseline is formulated as:

$$-\sum_{m_1=1}^{k} KL(L(S_{m_1})||L(X_{m_1})) + \left(\frac{1}{k-1}\sum_{m_2\in[1,k],m_1\neq m_2} KL(L(S_{m_1})||L(X_{m_2}))\right)$$

Our summary generation algorithm is to iteratively add excerpts to the summary in a greedy fashion, selecting the excerpt with the highest score in each iteration. Note that this approach only generates macro-level summaries, leaving us with the LexRank baseline for micro-level summaries.

4.3.3 Metrics

We will evaluate our summaries using a variant of the standard ROUGE evaluation metric (Lin, 2004).

Recall that we have two different evaluation sets – one that contains all of the reasons for each viewpoint, and one that consists only of aligned pairs of excerpts. Since the same excerpt may appear in multiple pairs, there would be significant redundancy in our reference summary if we were to include every pair. Thus, we will restrict a contrastive reference summary to exclude overlapping pairs, and we will have many reference sets for all possible combinations of pairs. There is only one reference set for the representativeness criterion.

Our reference summaries have a unique property in that the summaries have already been annotated with the prominence of the different reasons in the data. A good summary should capture the more prominent statements, so we will include this in our scoring function. We thus augment the basic ROUGE n-gram recall score by weighting the n-gram counts in the reference summary according to this percentage. This is a generalization of the standard ROUGE formula where this percentage would be uniform.

For evaluating the macro-level summaries, we will score the summaries for the two viewpoints separately, given a reference set Ref_i and a candidate summary C_i for a viewpoint $v = i$. The final score is a combination of the scores for both viewpoints, i.e. $S_{rep} = 0.5S(Ref_i, C_i) + 0.5S(Ref_j, C_j)$ where $S(Ref, C)$ is our ROUGE-based scoring metric. It would also be interesting to measure how well a viewpoint's summary matches the gold summary of the *opposite* viewpoint, which will give insights into how well the Comparative LexRank algorithm makes the two summaries similar to each other. We will measure this as the inverse of the above metric, i.e. $S_{opp} = 0.5S(Ref_i, C_j) + 0.5S(Ref_j, C_i)$.

Finally, to score the micro-level comparative summaries (recall that this gives explicitly-aligned pairs of excerpts), we will concatenate each pair (x_i, x_j) as a single excerpt, and use these as the excerpts in our reference and candidate summaries. The scoring function is then $S_p = S(Ref_{pairs}, C_{pairs})$. Note that we have multiple reference summaries for the

λ	S_{rep}**-1**	S_{opp}**-1**	S_{rep}**-2**	S_{opp}**-2**	S_p**-1**	S_p**-2**
0.0	**.425**	**.416**	.083	.060	.309	.036
0.2	**.410**	**.423**	.082	**.065**	.285	.044
0.5	**.419**	**.434**	.085	**.072**	**.386**	.044
0.8	**.410**	.324	**.095**	.028	**.367**	**.062**
1.0	.354	.240	.070	.006	.322	.057
MB	.362	.246	.089	.003		
MC	.347	.350	.054	.059		

Table 3: Our evaluation scores for various values of λ. Smaller values of λ favor greater contrastiveness. Note that $\lambda = 1$ should be considered a baseline, because at this value the algorithm ignores the contrastiveness and it becomes a standard summarization problem. MB and MC refer to our model-based baselines described in Subsection 4.3.2. Bold scores are significant over all baselines according to a paired t-test.

micro-level evaluation due to overlapping pairs in the evaluation set. In this case, the ROUGE score is defined as the maximum score among all possible reference summaries (Lin, 2004).

We measure both unigram (removing stop words, denoted S-1) and bigram (retaining stop words, denoted S-2) recall, stemming words in all cases.

4.3.4 Evaluation Results

In order to evaluate our Comparative LexRank algorithm by itself, in this subsection we will not use the output of TAM as part of our summarization input, and will assign excerpts fixed values of $P(v|x) = 1$ for the correct label and 0 otherwise. We constructed our sentence vectors with unigrams (removing stop words) and no IDF weighting.

We set the PageRank damping factor (Erkan and Radev, 2004) to 0.01 and tried combinations of the redundancy threshold $\delta \in \{0.01, 0.05, 0.1, 0.2\}$ with different values of λ, the parameter which controls the level of contrastiveness. For each value of λ, we optimized δ on the original data set according to $S_{rep} \times S_{opp}$ so that we can directly compare these scores, and then we tuned δ separately for S_p. The summary length is 6 excerpts. To obtain more robust results, we repeated the experiment 100 times on random half-size subsets of our data. The scores shown in Table 3 are averaged across these trials.

In general, increasing λ increases S_{rep}, which suggests that tuning λ behaves as expected, and high- and mid-range λ values indeed produce summaries where the summaries of the two viewpoints are more similar to each other. Similarly, mid-range λ values produce substantially higher values of S_p-1, the unigram ROUGE scores for the micro contrastive summary, although there is not a large difference between the bigram scores. An example of our micro-level output is shown in Table 4.

As for our model-based baseline, we show results for both the basic algorithm (denoted MB) in addition to the contrastive modification (denoted MC). We see that the contrastive modification behaves as expected and produces much higher scores for S_{opp}, however, this method does not outperform our LexRank algorithm. It is interesting to note that in almost all cases where a contrastive objective is introduced, the scores for the opposite viewpoint S_{opp} increase without decreasing the S_{rep} scores, suggesting that contrastiveness can be introduced into a multi-view summarization problem without diminishing the overall quality of the summary. It is admittedly difficult to make generalizations about these methods from experiments with only one data set, but we have at least some evidence that our algorithm works as intended.

4.4 Unsupervised Summarization

So far we have focused on evaluating our viewpoint clustering models and our multi-view summarization algorithms separately. We will finally show how these two stages might work in tandem in unsupervised summarization of the Bitterlemons corpus.

Without a gold set, it is difficult to perform an extensive automatic evaluation as we did with the healthcare data. Instead we will perform a simple qualitative evaluation to see if the algorithm appears to achieve its goal. Thus, we asked 8 people to guess if each viewpoint's summary was written by Israeli or Palestinian authors. To diversify the summaries, for each annotator we randomly split each summary into two equal-sized subsets of the sentence set. Thus each person was asked to label four different summaries, which were presented in a random order. If humans can correctly identify the viewpoints, then this would suggest both that the TAM accurately clustered documents by viewpoint and the summarization algorithm is selecting sentences that coherently represent the viewpoints.

We first ran TAM on our data using the same procedure and parameters as in Subsection 4.2 using the *full-tuple* features. We repeated this 10 times and used the model that gave the highest data likelihood as our model for summarization input. We then gen-

For the Healthcare Bill	*Against the Healthcare Bill*
the government already provides half of the healthcare dollars in the united states [...] [they] might as well spend their dollars smarter	government is too much involvement.
my kids are uninsured.	a lot of people will be getting it that should be getting it on their own, and my kids will be paying a lot of taxes.
so everybody would have it and afford it.	we cannot afford it.
because of my family.	i don't know enough about it and i don't know where exactly it's going to put my family.
because i have no health insurance and i need it.	because i have health insurance.
cost of healthcare is so high.	high costs.

Table 4: An example of our *micro-level* contrastive summarization output on the healthcare data, using $\delta = 0.05$ and $\lambda = 0.5$.

erated macro contrastive summaries of our data for the two viewpoints with 6 sentences per viewpoint. We used unigram sentence vectors with IDF weighting. We used $\lambda = 0.5$ and $\delta = 0.1$, which gave the highest score at this λ value on the healthcare data.

Only one of these sentences was clustered incorrectly by TAM. The human judges correctly labeled 78% of the summary sets, suggesting that our system accurately selected some sentences that could be recognized as belonging to the viewpoints, but is not perfect. Unsupervised micro-level summaries were less coherent. Many of the sentences are mislabeled, and the ones that are correctly labeled are not representative of the collection.

This is not surprising, and indeed exposes the challenge inherent in our problem definition: clustering documents based on similarity and then highlighting sentences with high similarity but opposite cluster membership are almost conflicting objectives for an unsupervised learner. Such contrastive pairs are perhaps the most difficult data points to model. A good test of a viewpoint model may be whether it can capture the nuanced properties of the viewpoints needed to contrast them at the micro level.

5 Discussion

The properties of the text which we attempt to summarize in our work are related to the concept of *framing* from political science (Chong and Druckman, 2010), which is defined as "an interpretation or evaluation of an issue, event, or person that emphasizes certain of its features or consequences" focusing on "certain features and implications of the issue – rather than others." For example, someone in favor of the healthcare bill might focus on the benefits and someone against the bill might focus on the cost.

However, our approach is different in that our contrastive objective encourages the summaries to include each point as addressed by all viewpoints, rather than each viewpoint selectively emphasizing only certain points. In a sense, this makes our summary more like a live debate, where one side must directly respond to a point raised by the other side. For example, someone in favor of healthcare reform might cite the high cost of the current system, but someone against this might counter-argue that the proposed system in the new bill has its own high costs (as seen in the last row of Table 4). The idea is to show how both sides address the same issues.

Thus, we can say that we are summarizing the key arguments/issues/points from different opinions. Futhermore, our models and algorithms are defined very generally, and while we tested their viability in the domain of political opinion, they may also be useful for many other comparative tasks.

In conclusion, we have presented steps toward a two-stage system that can automatically extract and summarize viewpoints in opinionated text. First, we have shown that accuracy of clustering documents by viewpoint can be enhanced by using simple but rich dependency features. This can be done within the framework of existing probabilistic topic models without altering the models simply by using a "bag of features" representation of documents.

Second, we have introduced *Comparative LexRank*, an extension of the LexRank algorithm that aims to generate contrastive summaries both at the macro and micro level. The algorithm presented is general enough that it can be applied to any number of viewpoints, and can accomodate input where the viewpoints are either given fixed labels, or given probabilistic assignments. The tradeoff between contrast and representation can flexibly be tuned to an application's needs.

References

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Samuel Brody and Noemie Elhadad. 2010. An unsupervised aspect-sentiment model for online reviews. In *NAACL '10*.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR '98*, pages 335–336.

Dennis Chong and James N. Druckman. 2010. Identifying frames in political news. In Erik P. Bucy and R. Lance Holbert, editors, *Sourcebook for Political Communication Research: Methods, Measures, and Analytical Techniques*. Routledge.

Cindy Chung and James W. Pennebaker. 2007. The psychological function of function words. *Social Communication: Frontiers of Social Psychology*, pages 343–359.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479.

Stephan Greene and Philip Resnik. 2009. More than words: syntactic packaging and implicit sentiment. In *NAACL '09*, pages 503–511.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *NAACL '09*, pages 362–370.

Sanda Harabagiu, Andrew Hickl, and Finley Lacatusu. 2006. Negation, contrast and contradiction in text processing.

Minqing Hu and Bing Liu. 2004. Mining opinion features in customer reviews. In *Proceedings of AAAI*, pages 755–760.

Minqing Hu and Bing Liu. 2006. Opinion extraction and summarization on the Web. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-2006), Nectar Paper Track*, Boston, MA.

Jeffrey M. Jones. 2010. "in u.s., 45% favor, 48% oppose obama healthcare plan", March.

Mahesh Joshi and Carolyn Penstein Rosé. 2009. Generalizing dependency features for opinion mining. In *ACL-IJCNLP '09: Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 313–316.

Hyun Duk Kim and ChengXiang Zhai. 2009. Generating comparative summaries of contradictory opinions in text. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, pages 385–394, New York, NY, USA. ACM.

Kevin Lerman and Ryan McDonald. 2009. Contrastive summarization: an experiment with consumer reviews. In *NAACL '09*, pages 113–116, Morristown, NJ, USA. Association for Computational Linguistics.

Wei-Hao Lin, Theresa Wilson, Janyce Wiebe, and Alexander Hauptmann. 2006. Which side are you on?: identifying perspectives at the document and sentence levels. In *CoNLL-X '06: Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 109–116.

Wei-Hao Lin, Eric Xing, and Alexander Hauptmann. 2008. A joint topic and perspective model for ideological discourse. In *ECML PKDD '08: Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases - Part II*, pages 17–32, Berlin, Heidelberg. Springer-Verlag.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In Stan Szpakowicz Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain, July.

Bing Liu, Minqing Hu, and Junsheng Cheng. 2005. Opinion observer: analyzing and comparing opinions on the web. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 342–351, New York, NY, USA. ACM Press.

Marie-Catherine De Marneffe and Christopher Manning. 2008. Stanford typed dependencies manual. Technical report, Stanford University.

Marie-Catherine De Marneffe, Anna Rafferty, and Christopher Manning. 2008. Finding contradictions in text. In *Proceedings of the Association for Computational Linguistics Conference (ACL)*.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *ECIR'07: Proceedings of the 29th European conference on IR research*, pages 557–564, Berlin, Heidelberg. Springer-Verlag.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1998. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project.

Michael Paul and Roxana Girju. 2010. A two-dimensional topic-aspect model for discovering multi-faceted topics. In *AAAI-2010: Twenty-Fourth Conference on Artificial Intelligence*.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 347–354.

Li Zhuang, Feng Jing, Xiao-yan Zhu, and Lei Zhang. 2006. Movie review mining and summarization. In *Proceedings of the ACM SIGIR Conference on Information and Knowledge Management (CIKM)*.

Automatically Producing Plot Unit Representations for Narrative Text

Amit Goyal
Dept. of Computer Science
University of Maryland
College Park, MD 20742
amit@umiacs.umd.edu

Ellen Riloff
School of Computing
University of Utah
Salt Lake City, UT 84112
riloff@cs.utah.edu

Hal Daumé III
Dept. of Computer Science
University of Maryland
College Park, MD 20742
hal@umiacs.umd.edu

Abstract

In the 1980s, plot units were proposed as a conceptual knowledge structure for representing and summarizing narrative stories. Our research explores whether current NLP technology can be used to automatically produce plot unit representations for narrative text. We create a system called AESOP that exploits a variety of existing resources to identify affect states and applies "projection rules" to map the affect states onto the characters in a story. We also use corpus-based techniques to generate a new type of affect knowledge base: verbs that impart positive or negative states onto their patients (e.g., being eaten is an undesirable state, but being fed is a desirable state). We harvest these "patient polarity verbs" from a Web corpus using two techniques: co-occurrence with Evil/Kind Agent patterns, and bootstrapping over conjunctions of verbs. We evaluate the plot unit representations produced by our system on a small collection of Aesop's fables.

1 Introduction

In the 1980s, plot units (Lehnert, 1981) were proposed as a knowledge structure for representing narrative stories and generating summaries. Plot units are fundamentally different from the story representations that preceded them because they focus on the affect states of characters and the tensions between them as the driving force behind interesting and cohesive stories. Plot units were used in narrative summarization studies, both in computer science and psychology (Lehnert et al., 1981), but previous computational models of plot units relied on tremendous amounts of manual knowledge engineering.

The last few decades have seen tremendous advances in NLP and the emergence of many resources that could be useful for plot unit analysis. So we embarked on a project to see whether plot unit representations can be generated automatically using current NLP technology. We created a system called AESOP that uses a variety of resources to identify words that correspond to positive, negative, and mental *affect states*. AESOP uses *affect projection rules* to map the affect states onto the characters in the story based on verb argument structure. Additionally, affect states are inferred based on syntactic properties, and causal and cross-character links are created using simple heuristics.

Affect states often arise from actions that produce good or bad states for the character that is acted upon. For example, *"the cat ate the mouse"* produces a negative state for the mouse because being eaten is bad. Similarly, *"the man fed the dog"* produces a positive state for the dog because being fed is generally good. Knowledge about the effects of actions (i.e., state changes) on patients is not readily available in existing semantic resources. We create a new type of lexicon consisting of *patient polarity verbs (PPVs)* that impart positive or negative states on their patients. These verbs reflect world knowledge about desirable/undesirable states for animate beings; for example, being *fed*, *paid* or *adopted* are generally desirable states, while being *eaten*, *chased* or *hospitalized* are generally undesirable states.

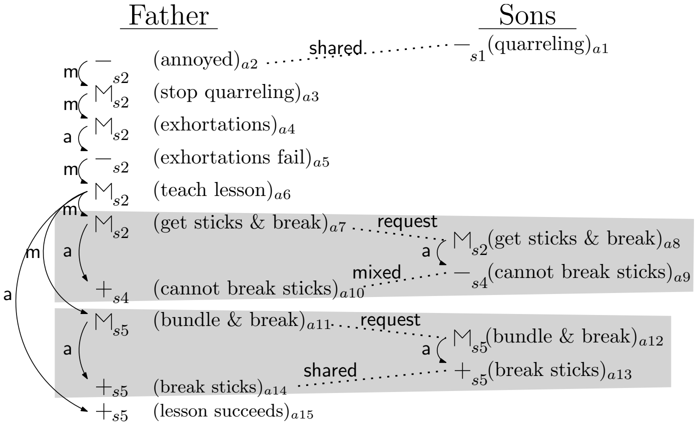
We automatically generate a lexicon of "patient polarity verbs" from a Web corpus using two tech-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 77–86,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

The Father and His Sons

(s1) A father had a family of sons who were perpetually quarreling among themselves. (s2) When he failed to heal their disputes by his exhortations, he determined to give them a practical illustration of the evils of disunion; and for this purpose he one day told them to bring him a bundle of sticks. (s3) When they had done so, he placed the faggot into the hands of each of them in succession, and ordered them to break it in pieces. (s4) They tried with all their strength, and were not able to do it. (s5) He next opened the faggot, took the sticks separately, one by one, and again put them into his sons' hands, upon which they broke them easily. (s6) He then addressed them in these words: "My sons, if you are of one mind, and unite to assist each other, you will be as this faggot, uninjured by all the attempts of your enemies; but if you are divided among yourselves, you will be broken as easily as these sticks."

(a) "Father and Sons" Fable



Father
Sons
$-_{s1}$(quarreling)$_{a1}$
$-_{s2}$ (annoyed)$_{a2}$
shared
m
M_{s2} (stop quarreling)$_{a3}$
m
M_{s2} (exhortations)$_{a4}$
a
$-_{s2}$ (exhortations fail)$_{a5}$
m
M_{s2} (teach lesson)$_{a6}$
m
M_{s2} (get sticks & break)$_{a7}$
request
M_{s2}(get sticks & break)$_{a8}$
a
a
mixed
$-_{s4}$(cannot break sticks)$_{a9}$
m
$+_{s4}$ (cannot break sticks)$_{a10}$
a
M_{s5} (bundle & break)$_{a11}$
request
M_{s5}(bundle & break)$_{a12}$
a
a
shared
$+_{s5}$(break sticks)$_{a13}$
$+_{s5}$ (break sticks)$_{a14}$
$+_{s5}$ (lesson succeeds)$_{a15}$

(b) Plot Unit Analysis for "Father and Sons" Fable

Figure 1: Sample Fable and Plot Unit Representation

niques: patterns that identify co-occurrence with stereotypically evil or kind agents, and a bootstrapping algorithm that learns from conjunctions of verbs. We evaluate the plot unit representations produced by our system on a small collection of fables.

2 Overview of Plot Units

Plot unit structures consist of *affect states* for each character, and links defining the relationships between them. Plot units include three types of affect states: positive (+), negative (-), and mental (M). Affect states can be connected by *causal links* and *cross-character* links, which explain how the narrative hangs together. Causal links exist between affect states for the same character and have four types: motivation (m), actualization (a), termination (t) and equivalence (e). Cross-character links indicate that a single event affects multiple characters. For instance, if one character *requests* something of another, then each character is assigned an M state and a cross-character link connects the states.

To see a concrete example of a plot unit representation, a short fable, "The Father and His Sons," is shown in Figure 1(a) and our annotation of its plot unit structure is shown in Figure 1(b). In this fable, there are two characters, the "Father" and (collectively) the "Sons", who go through a series of affect states depicted chronologically in the two columns.

The first affect state ($a1$) is produced from sentence #1 ($s1$) and is a negative state for the sons because they are quarreling. This state is *shared* by the father (via a cross-character link) who has a negative annoyance state ($a2$). The father decides that he wants to stop the sons from quarreling, which is a mental event ($a3$). The causal link from $a2$ to $a3$ with an m label indicates that his annoyed state "motivated" this decision. His first attempt is by exhortations ($a4$). The first M ($a3$) is connected to the second M ($a4$) with an m (motivation) link, which represents subgoaling. The father's overall goal is to stop the quarreling ($a3$), and to do so he creates a subgoal of exhorting the sons to stop ($a4$). The exhortations fail, which produces a negative state ($a5$) for the father. The a causal link indicates an "actualization", representing the failure of his plan ($a4$).

This failure motivates a new subgoal: teach the sons a lesson ($a6$). At a high level, this subgoal has two parts, indicated by the two gray regions ($a7 - a10$ and $a11 - a14$). The first gray region begins with a cross-character link (M to M), which indicates a request (in this case, to break a bundle of sticks). The sons fail at this, which upsets them ($a9$) but pleases the father ($a10$). The second gray region depicts the second part of the father's subgoal; he makes a second request ($a11$ to $a12$) to separate the bundle and break the sticks, which the sons successfully do, making them happy ($a13$) and the father happy ($a14$) as well. This latter structure (the second gray region) is an HONORED REQUEST plot unit structure. At the end, the father's plan succeeds ($a15$) which is an actualization (a link) of his goal to teach the sons a lesson ($a6$).

3 Where Do Affect States Come From?

We briefly overview the variety of situations that can be represented by affect states in plot units.

Direct Expressions of Emotion: Affect states can correspond to positive/negative emotional states, as have been studied in the realm of sentiment analysis. For example, *"Max was disappointed"* produces a negative affect state for Max, and *"Max was pleased"* produces a positive affect state for Max.

Situational Affect States: Positive and negative affect states can represent good and bad situational states that characters find themselves in. These states do not represent emotion, but indicate whether a situation (state) is good or bad for a character based on world knowledge. e.g., *"The wolf had a bone stuck in his throat."* produces a negative affect state for the wolf. Similarly, *"The woman recovered her sight."* produces a positive affect state for the woman.

Plans and Goals: The existence of a plan or goal is represented as a mental state (M). Plans and goals can be difficult to detect automatically and can be revealed in many ways, such as:

- **Direct expressions of plans/goals:** a plan/goal may be explicitly stated (e.g., *"John wants food"*).
- **Speech acts:** a plan or goal may be revealed through a speech act. For example, *"the wolf asked an eagle to extract the bone"* is a directive speech act that indicates the wolf's plan to resolve its negative state (having a bone stuck). This example illustrates how a negative state (bone stuck) can motivate a mental state (plan). When a speech act involves multiple characters, it produces multiple mental states.
- **Inferred plans/goals:** plans and goals are sometimes inferred from actions. e.g., *"the lion hunted deer"* implies that the lion has a plan to obtain food. Similarly, *"the serpent spat poison at John"* implies that the serpent wants to kill John.
- **Plan/Goal completion:** Plans and goals produce +/- affect states when they succeed or fail. For example, if the eagle successfully extracts the bone from the wolf's throat, then both the wolf and the eagle will have positive affect states because both were successful in their respective goals.

We observed that situational and plan/goal states often originate from an action. When a character is acted upon (the *patient* of a verb), then the character may be in a positive or negative state depending upon whether the action was good or bad for them based on world knowledge. For example, being *fed*, *paid* or *adopted* is generally desirable, but being *chased*, *eaten*, or *hospitalized* is usually undesirable. Consequently, we decided to create a lexicon of *patient polarity verbs* that produce positive or negative states for their patients. In Section 4.2, we present two methods for automatically harvesting these verbs from a Web corpus.

4 AESOP: Automatically Generating Plot Unit Representations

Our system, AESOP, automatically creates plot unit representations for narrative text. AESOP has four main steps: affect state recognition, character identification, affect state projection, and link creation. During affect state recognition, AESOP identifies words that may be associated with positive, negative, and mental states. AESOP then identifies the main characters in the story and applies *affect projection rules* to map the affect states onto these characters. During this process, some additional affect states are inferred based on verb argument structure. Finally, AESOP creates cross-character links and causal links between affect states. We also present two corpus-based methods to automatically produce a new resource for affect state recognition: a *patient polarity verb lexicon*.

4.1 Plot Unit Creation

4.1.1 Recognizing Affect States

The basic building blocks of plot units are *affect states* which come in three flavors: positive, negative, and mental. In recent years, many publicly available resources have been created for sentiment analysis and other types of semantic knowledge. We considered a wide variety of resources and ultimately decided to experiment with five resources that most closely matched our needs:

- FrameNet (Baker et al., 1998): We manually identified 87 frame classes that seem to be associated with affect: 43 mental classes (e.g., COMMUNICATION and NEEDING), 22 positive classes (e.g., ACCOMPLISHMENT and SUPPORTING), and 22 negative classes (e.g., CAUSE HARM and PROHIBIT-

ING). We use the verbs listed for these classes to produce M, +, and - affect states.

• MPQA Lexicon (Wilson et al., 2005b): We used the words listed as having positive or negative sentiment polarity to produce +/- states, when they occur with the designated part-of-speech.

• OpinionFinder (Wilson et al., 2005a) (Version 1.4) : We used the +/- labels assigned by its contextual polarity classifier (Wilson et al., 2005b) to create +/- states and the MPQASD tags produced by its Direct Subjective and Speech Event Identifier (Choi et al., 2006) to produce mental (M) states.

• Semantic Orientation Lexicon (Takamura et al., 2005): We used the words listed as having positive or negative polarity to produce +/- affect states, when they occur with the designated part-of-speech.

• Speech Act Verbs: We used 228 speech act verbs from (Wierzbicka, 1987) to produce M states.

4.1.2 Identifying the Characters

For the purposes of this work, we made two simplifying assumptions: (1) There are only two characters per fable[1], and (2) Both characters are mentioned in the fable's title. The problem of coreference resolution for fables is somewhat different than for other genres, primarily because the characters are often animals (e.g., *he=owl*). So we hand-crafted a simple rule-based coreference system. First, we apply heuristics to determine number and gender based on word lists, WordNet (Miller, 1990) and part-of-speech tags. If no determination of a character's gender or number can be made, we employ a process of elimination. Given the two character assumption, if one character is known to be male, but there are female pronouns in the fable, then the other character is assumed to be female. The same is done for number agreement. Finally, if there is only one character between a pronoun and the beginning of a document, then we resolve the pronoun with that character and the character assumes the gender and number of the pronoun. Lastly, WordNet provides some additional resolutions by exploiting hypernym relations, for instance, linking *peasant* with *man*.

4.1.3 Mapping Affect States onto Characters

Plot unit representations are not just a set of affect states, but they are structures that capture the chronological ordering of states for each character as the narrative progresses. Consequently, every affect state needs to be attributed to a character. Since most plots revolve around events, we use verb argument structure as the primary means for projecting affect states onto characters.

We developed four *affect projection rules* that orchestrate how affect states are assigned to the characters. We used the Sundance parser (Riloff and Phillips, 2004) to produce a shallow parse of each sentence, which includes syntactic chunking, clause segmentation, and active/passive voice recognition. We normalized the verb phrases with respect to active/passive voice to simplify the rules. We made the assumption that the Subject of the VP is its AGENT and the Direct Object of the VP is its PATIENT.[2] The rules only project affect states onto AGENTS and PATIENTS that refer to a character in the story. The four projection rules are presented below.

1. AGENT **VP** : This rule applies when the VP has no PATIENT or the PATIENT corefers with the AGENT. All affect tags assigned to the VP are projected onto the AGENT. Example: *"Mary **laughed** (+)"* projects a + affect state onto Mary.

2. **VP** PATIENT : This rule applies when the VP has no agent, which is common in passive voice constructions. All affect tags assigned to the VP are projected onto the PATIENT. Example: *"John was **rewarded** (+)*, projects a + affect state onto John.

3. AGENT **VP** PATIENT : This rules applies when both an AGENT and PATIENT are present, do not corefer, and at least one of them is a character. If the PATIENT is a character, then all affect tags associated with the VP are projected onto the PATIENT. If the AGENT is a character and the VP has an M tag, then we also project an M tag onto the AGENT (representing a shared, cross-character mental state).

4. AGENT **VERB1** to **VERB2** PATIENT : This rule has two cases: (a) If the AGENT and PATIENT refer to the same character, then we apply Rule #1. Example: *"Bo decided to teach himself..."* (b) If the AGENT and PATIENT are different, then we apply Rule #1 to **VERB1** and Rule #2 to **VERB2**.

Finally, if an adverb or adjectival phrase has affect, then that affect is mapped onto the preceding VP and the rules above are applied. For all of the

[1] We only selected fables that had two main characters.

[2] This is not always correct, but worked ok in our fables.

rules, if a clause contains a negation word, then we flip the polarity of all words in that clause.

4.1.4 Inferring Affect States

Recognizing plans and goals depends on world knowledge and inference, and is beyond the scope of this paper. However, we identified two cases where affect states often can be inferred based on syntactic properties. The first case involves verb phrases (VPs) that have both an AGENT and PATIENT, which corresponds to projection rule #3. If the VP has polarity, then rule #3 assigns that polarity to the PATIENT, not the AGENT. For example, *"John killed Paul"* imparts negative polarity on Paul, but not necessarily on John. Unless we are told otherwise, one assumes that John *intentionally* killed Paul, and so in a sense, John accomplished his goal. Consequently, this action should produce a positive affect state for John. We capture this notion of accomplishment as a side effect of projection rule #3: if the VP has +/- polarity, then we produce an *inferred positive state* for the AGENT.

The second case involves infinitive verb phrases of the form: "AGENT VERB1 TO VERB2 PATIENT" (e.g., *"Susan tried to warn Mary"*). The infinitive VP construction suggests that the AGENT has a goal or plan that is being put into motion (e.g., *tried to, wanted to, attempted to, hoped to*, etc.). To capture this intuition, in rule #4 if VERB1 does not already have an affect state assigned to it then we produce an *inferred mental state* for the AGENT.

4.1.5 Causal and Cross-Character Links

Our research is focused primarily on creating affect states for characters, but plot unit structures also include *cross-character links* to connect states that are shared across characters and *causal links* between states for a single character. As an initial attempt to create complete plot units, AESOP produces links using simple heuristics. A cross-character link is created when two characters in a clause have affect states that originated from the same word. A causal link is created between each pair of (chronologically) consecutive affect states for the same character. Currently, AESOP only produces forward causal links (*motivation (m)*, *actualization (a)*) and does not produce backward causal links (*equivalence (e)*, *termination (t)*). For forward links, the causal syntax only allows for five cases: $M \xrightarrow{m} M$, $+ \xrightarrow{m} M$, $- \xrightarrow{m} M$, $M \xrightarrow{a} +$, $M \xrightarrow{a} -$. So when AESOP produces a causal link between two affect states, the order and types of the two states uniquely determine which label it gets (m or a).

4.2 Generating PPV Lexicons

During the course of this research, we identified a gap in currently available knowledge: we are not aware of existing resources that identify verbs which produce a desirable/undesirable state for their patients even though the verb itself does not carry polarity. For example, the verb *eat* describes an action that is generally neutral, but being eaten is clearly an undesirable state. Similarly, the verb *fed* does not have polarity, but being fed is a desirable state for the patient. In the following sections, we try to fill this gap by using corpus-based techniques to automatically acquire a *Patient Polarity Verb (PPV) Lexicon*.

4.2.1 PPV Harvesting with Evil/Kind Agents

The key idea behind our first approach is to identify verbs that frequently occur with evil or kind agents. Our intuition was that an "evil" agent will typically perform actions that are bad for the patient, while a "kind" agent will typically perform actions that are good for the patient.

We manually identified 40 stereotypically evil agent words, such as *monster*, *villain*, *terrorist*, and *murderer*, and 40 stereotypically kind agent words, such as *hero*, *angel*, *benefactor*, and *rescuer*. We searched the Google Web $1T$ N-gram corpus to identify verbs that co-occur with these words as probable agents. For each agent term, we applied the pattern "* by [a,an,the] AGENT" and extracted the matching N-grams. Then we applied a part-of-speech tagger to each N-gram and saved the words that were tagged as verbs (i.e., the words in the * position).[3] This process produced 811 negative (evil agent) PPVs and 1362 positive (kind agent) PPVs.

4.2.2 PPV Bootstrapping over Conjunctions

Our second approach for acquiring PPVs is based on an observation from sentiment analysis research that conjoined adjectives typically have the same polarity (e.g. (Hatzivassiloglou and McKeown, 1997)).

[3] The POS tagging quality is undoubtedly lower than if tagging complete sentences but it seemed reasonable.

Our hypothesis was that conjoined verbs often share the same polarity as well (e.g., *"abducted and killed"* or *"rescued and rehabilitated"*). We exploit this idea inside a bootstrapping algorithm to iteratively learn verbs that co-occur in conjunctions.

Bootstrapping begins with 10 negative and 10 positive PPV seeds. First, we extracted triples of the form *"w1 and w2"* from the Google Web $1T$ N-gram corpus that had frequency ≥ 100 and were lower case. We separated each conjunction into two parts: a primary VERB ("$w1$") and a CONTEXT ("$and\ w2$"), and created a copy of the conjunction with the roles of $w1$ and $w2$ reversed. For example, *"rescued and adopted"* produces:

VERB="rescued" CONTEXT="and adopted"
VERB="adopted" CONTEXT="and rescued"

Next, we applied the Basilisk bootstrapping algorithm (Thelen and Riloff, 2002) to learn PPVs. Basilisk identifies semantically similar words based on their co-occurrence with seeds in contextual patterns. Basilisk was originally designed for semantic class induction using lexico-syntactic patterns, but has also been used to learn subjective and objective nouns (Riloff et al., 2003).

Basilisk first identifies the pattern contexts that are most strongly associated with the seed words. Words that occur in those contexts are labeled as *candidates* and scored based on the strength of their contexts. The top 5 candidates are selected and the bootstrapping process repeats. Basilisk produces a *lexicon* of learned words as well as a ranked list of pattern contexts. Since we bootstrapped over verb conjunctions, we also extracted new PPVs from the contexts. We ran the bootstrapping process to create a lexicon of 500 words, and we collected verbs from the top 500 contexts as well.

5 Evaluation

Plot unit analysis of narrative text is enormously complex – the idea of creating gold standard plot unit annotations seemed like a monumental task. So we began with relatively simple and constrained texts that seemed appropriate: fables. Fables have two desirable attributes: (1) they have a small cast of characters, and (2) they typically revolve around a moral, which is exemplified by a short and concise plot. Even so, fables are challenging for NLP due to anthropomorphic characters, flowery language, and sometimes archaic vocabulary.

We collected 34 Aesop's fables from a web site[4], choosing fables that have a true plot (some only contain quotes) and exactly two characters. We divided them into a development set of 11 stories, a tuning set of 8 stories, and a test set of 15 stories.

Creating a gold standard was itself a substantial undertaking, and training non-experts to produce them did not seem feasible in the short term. So the authors discussed and iteratively refined manual annotations for the development and tuning sets until we produced similar results and had a common understanding of the task. Then two authors independently created annotations for the test set, and a third author adjudicated the differences.

5.1 Evaluation Procedure

For evaluation, we used recall (R), precision (P), and F-measure (F). In our gold standard, each affect state is annotated with the set of clauses that could legitimately produce it. In most cases (75%), we were able to ascribe the existence of a state to precisely one clause. During evaluation, the system-produced affect states must be generated from the correct clause. However, for affect states that could be ascribed to multiple clauses in a sentence, the evaluation was done at the sentence level. In this case, the system-produced affect state must come from the sentence that contains one of those clauses.

Coreference resolution is far from perfect, so we created gold standard coreference annotations for our fables and used them for most of our experiments. This allowed us to evaluate our approach without coreference mistakes factoring in. In Section 5.5, we re-evaluate our final results using automatic coreference resolution.

5.2 Evaluation of Affect States using External Resources

Our first set of experiments evaluates the quality of the affect states produced by AESOP using only the external resources. The top half of Table 1 shows the results for each resource independently. FrameNet produced the best results, yielding much higher recall than any other resource. The bottom half of Ta-

[4] www.pacificnet.net/~johnr/aesop/

Affect State	**M** (59)			**+** (47)			**-** (37)			**All** (143)		
Resource(s)	*R*	*P*	*F*	*R*	*P*	*F*	*R*	*P*	*F*	*R*	*P*	*F*
FrameNet	.49	.51	.50	.17	.57	.26	.14	.42	.21	.29	.51	.37
MPQA Lexicon	.07	.50	.12	.21	.24	.22	.22	.38	.28	.15	.31	.20
OpinionFinder	.42	.40	.41	.00	.00	.00	.03	.17	.05	.18	.35	.24
Semantic Orientation Lexicon	.07	.44	.12	.17	.40	.24	.08	.38	.13	.10	.41	.16
Speech Act Verbs	.36	.53	.43	.00	.00	.00	.00	.00	.00	.15	**.53**	.23
FrameNet+MPQA Lexicon	.44	.52	.48	.30	.28	.29	.27	.38	.32	**.35**	.40	.37
FrameNet+OpinionFinder	.53	.39	.45	.17	.38	.23	.16	.33	.22	.31	.38	.34
FrameNet+Semantic Orientation Lexicon	.49	.51	.50	.26	.36	.30	.22	.42	.29	.34	.45	**.39**
FrameNet+Speech Act Verbs	.51	.48	.49	.17	.57	.26	.14	.42	.21	.30	.49	.37

Table 1: Evaluation results for AESOP using external resources. The # in parentheses is the # of gold affect states.

Affect State	**M** (59)			**+** (47)			**-** (37)			**All** (143)		
Resource(s)	*R*	*P*	*F*	*R*	*P*	*F*	*R*	*P*	*F*	*R*	*P*	*F*
- Evil Agent PPVs	.07	.50	.12	.21	.40	.28	.46	.46	.46	.22	.44	.29
- Neg Basilisk PPVs	.07	.44	.12	.11	.45	.18	.24	.45	.31	.13	.45	.20
- Evil Agent and Neg Basilisk PPVs	.05	.43	.09	.21	.38	.27	.46	.40	.43	.21	.39	.27
+ Kind Agent PPVs (θ>1)	.03	.33	.06	.28	.17	.21	.00	.00	.00	.10	.19	.13
+ Pos Basilisk PPVs	.08	.56	.14	.02	.12	.03	.03	1.00	.06	.05	.39	.09
FrameNet+SOLex+EvilAgentPPVs	.49	.54	.51	.30	.38	.34	.46	.42	.44	.42	.46	.44
FrameNet+EvilAgentPPVs	.49	.54	.51	.28	.45	.35	.46	.46	.46	.41	.49	**.45**
FrameNet+EvilAgentPPVs+PosBasiliskPPVs	.49	.53	.51	.30	.41	.35	.49	.49	.49	**.43**	.48	**.45**

Table 2: Evaluation results for AESOP with PPVs. The # in parentheses is the # of gold affect states.

ble 1 shows the results when combining FrameNet with other resources. In terms of F score, the only additive benefit came from the Semantic Orientation Lexicon, which produced a better balance of recall and precision and an F score gain of +2.

5.3 Evaluation of Affect States using PPVs

Our second set of experiments evaluates the quality of the automatically generated PPV lexicons. The top portion of Table 2 shows the results for the negative PPVs. The PPVs harvested by the Evil Agent patterns produced the best results, yielding recall and precision of .46 for negative states. Note that M and + states are also generated from the negative PPVs because they are inferred during affect projection (Section 4.1.4). The polarity of a negative PPV can also be flipped by negation to produce a + state.

Basilisk's negative PPVs achieved similar precision but lower recall. We see no additional recall and some precision loss when the Evil Agent and Basilisk PPV lists are combined. The precision drop is likely due to redundancy, which creates spurious affect states. If two different words have negative polarity but refer to the same event, then only one negative affect state should be generated. But AESOP will generate two affect states, so one will be spurious.

The middle section of Table 2 shows the results for the positive PPVs. Both positive PPV lexicons were of dubious quality, so we tried to extract a high-quality subset of each list. For the Kind Agent PPVs, we computed the ratio of the frequency of the verb with Evil Agents versus Kind Agents and only saved verbs with an Evil:Kind ratio (θ) > 1, which yielded 1203 PPVs. For the positive Basilisk PPVs, we used only the top 100 lexicon and top 100 context verbs, which yielded 164 unique verbs. The positive PPVs did generate several correct affect states (including a - state when a positive PPV was negated), but also many spurious states.

The bottom section of Table 2 shows the impact of the learned PPVs when combined with FrameNet and the Semantic Orientation Lexicon (SOLex). Adding the Evil Agent PPVs improved AESOP's F score from 39% to 44%, mainly due to a +8 recall gain. The recall of the - states increased from 22% to 46% with no loss of precision. Interestingly, if we remove SOLex and use only FrameNet with our PPVs, precision increases from 46% to 49% and recall only drops by -1. Finally, the last row of Table

2 shows that adding Basilisk's positive PPVs produces a small recall boost (+2) with a slight drop in precision (-1).

Evaluating the impact of PPVs on plot unit structures is an indirect way of assessing their quality because creating plot units involves many steps. Also, our test set is small so many verbs will never appear. To directly measure the quality of our PPVs, we recruited 3 people to manually review them. We developed annotation guidelines that instructed each annotator to judge whether a verb is generally *good* or *bad* for its patient, assuming the patient is animate. They assigned each verb to one of 6 categories: × (not a verb), 2 (always good), 1 (usually good), 0 (neutral, mixed, or requires inanimate patient), -1 (usually bad), -2 (always bad). Each annotator labeled 250 words: 50 words randomly sampled from each of our 4 PPV lexicons[5] (Evil Agent PPVs, Kind Agent PPVs, Positive Basilisk PPVs, and Negative Basilisk PPVs) plus 50 verbs labeled as neutral in the MPQA lexicon.

First, we measured agreement based on three groupings: *negative* (-2 and -1), *neutral* (0), or *positive* (1 and 2). We computed κ scores to measure inter-annotator agreement for each pair of annotators.[6], but the κ scores were relatively low because the annotators had trouble distinguishing the positive cases from the neutral ones. So we re-computed agreement using two groupings: *negative* (-2 and -1) and *not-negative* (0 through 2), and obtained κ scores of .69, .71, and .74. We concluded that people largely agree on whether a verb is bad for the patient, but they do not necessarily agree if a verb is good for the patient. One possible explanation is that many "bad" verbs represent physical harm or danger: these verbs are both plentiful and easy to recognize. In contrast, "good" verbs are often more abstract and open to interpretation (e.g., is being "envied" or "feared" a good thing?).

We used the labels produced by the two annotators with the highest κ score to measure the accuracy of our PPVs. Both the Evil Agent and Negative Basilisk PPVs were judged to be 72.5% accurate, averaged over the judges. The Kind Agent PPVs were only about 39% accurate, while the Positive Basilisk PPVs were nearly 50% accurate. These results are consistent with our impressions that the negative PPVs are of relatively high quality, while the positive PPVs are mixed. Some examples of learned PPVs that were not present in our other resources are:

- : censor, chase, fire, orphan, paralyze, scare, sue
+ : accommodate, harbor, nurse, obey, respect, value

5.4 Evaluation of Links

We represented each link as a 5-tuple ⟨*src-clause*, *src-state*, *tgt-clause*, *tgt-state*, *link-type*⟩, where source/target denotes the direction of the link, the source/target-states are the affect state type (+,-,M) and *link-type* is one of 3 types: actualization (a), motivation (m), or cross-character (xchar). A system-produced link is considered correct if *all* 5 elements of the tuple match the human annotation.

	Gold Aff States			*System Aff States*		
Links	*R*	*P*	*F*	*R*	*P*	*F*
xchar (56)	.79	.85	.82	.18	.43	.25
a (51)	.90	.94	.92	.04	.07	.05
m (26)	1.0	.57	.72	.15	.10	.12

Table 3: Link results; parentheses show # of gold links.

The second column of Table 3 shows the performance of AESOP when using gold standard affect states. Our simple heuristics for creating links work surprisingly well for xchar and a links when given perfect affect states. However, these heuristics produce relatively low precision for m links, albeit with 100% recall. This reveals that m links primarily do connect adjacent states, but we need to be more discriminating when connecting them. The third column of Table 3 shows the results when using system-generated affect states. We see that performance is much lower. This is not particularly surprising, since AESOP's F-score is 45%, so over half of the individual states are wrong, which means that less than a quarter of the pairs are correct. From that perspective, the xchar link performance is reasonable, but the causal a and m links need improvement.

5.5 Analysis

We performed additional experiments to evaluate some assumptions and components. First, we created a *Baseline* system that is identical to AESOP

[5] The top-ranked Evil/Kind Agent PPV lists ($\theta > 1$) which yields 1203 kind PPVs, and 477 evil PPVs, the top 164 positive Basilisk verbs, and the 678 (unique) negative Basilisk verbs.

[6] We discarded words labeled as not a verb.

except that it does not use the affect projection rules. Instead, it naively projects every affect state in a clause onto every character in that clause. The first two rows of the table below show that AESOP's precision is double the Baseline, with nearly the same recall. This illustrates the importance of the projection rules for mapping affect states onto characters.

	R	*P*	*F*
Baseline	.44	.24	.31
AESOP, gold coref	.43	.48	.45
AESOP, gold coref, infstates	.39	.48	.43
AESOP, auto coref, infstates	.24	.56	.34

Our gold standard includes *pure inference affect states* that are critical to the plot unit structure but come from world knowledge outside the story itself. Of 157 affect states in our test set, 14 were pure inference states. We ignored these states in our previous experiments because our system has no way to generate them. The third row of the table shows that including them lowers recall by -4. Generating pure inferences is an interesting challenge, but they seem to be a relatively small part of the problem.

The last row of the table shows AESOP's performance when we use our automated coreference resolver (Section 4.1.2) instead of gold standard coreference annotations. We see a -15 recall drop coupled with a +8 precision gain. We were initially puzzled by the precision gain but believe that it is primarily due to the handling of quotations. Our gold standard includes annotations for characters mentioned in quotations, but our automated coreference resolver ignores quotations. Most fables end with a moral, which is often a quote that may not mention the plot. Consequently, AESOP generates more spurious affect states from the quotations when using the gold standard annotations.

6 Related Work and Conclusions

Our research is the first effort to fully automate the creation of plot unit structures. Other preliminary work has begun to look at plot unit modelling for single character stories (Appling and Riedl, 2009). More generally, our work is related to research in narrative story understanding (e.g., (Elson and McKeown, 2009)), automatic affect state analysis (Alm, 2009), and automated learning of scripts (Schank and Abelson, 1977) and other conceptual knowledge structures (e.g., (Mooney and DeJong, 1985; Fujiki et al., 2003; Chambers and Jurafsky, 2008; Chambers and Jurafsky, 2009; Kasch and Oates, 2010)). Our work benefitted from prior research in creating semantic resources such as FrameNet (Baker et al., 1998) and sentiment lexicons and classifiers (e.g., (Takamura et al., 2005; Wilson et al., 2005b; Choi et al., 2006)). We showed that *affect projection rules* can effectively assign affect states to characters. This task is similar to, but not the same as, associating opinion words with their targets or topics (Kim and Hovy, 2006; Stoyanov and Cardie, 2008). Some aspects of affect state identification are closely related to Hopper and Thompson's (1980) theory of transitivity. In particular, their notions of *aspect* (has an action completed?), benefit and harm (how much does an object gain/lose from an action?) and volition (did the subject make a conscious choice to act?).

AESOP produces affect states with an F score of 45%. Identifying *positive* states appears to be more difficult than negative or mental states. Our system's biggest shortcoming currently seems to hinge around identifying plans and goals. This includes the M affect states that initiate plans, the +/- completion states, as well as their corresponding links. We suspect that the relatively low recall on positive affect states is due to our inability to accurately identify successful plan completions. Finally, these results are based on fables; plot unit analysis of other types of texts will pose additional challenges.

Acknowledgments

The authors gratefully acknowledge the support of Department of Homeland Security Grant N0014-07-1-0152, NSF grant IIS-0712764, and the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0172. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the DARPA, AFRL, or the U.S. government. Thanks to Peter Jensen, Emily Schlichter, and Clay Templeton for PPV annotations, Nathan Gilbert for help with the coreference resolver, and the anonymous reviewers for many helpful comments.

References

Cecilia Ovesdotter Alm. 2009. *Affect in Text and Speech*. VDM Verlag Dr. Mller.

D. Scott Appling and Mark O. Riedl. 2009. Representations for learning to summarize plots. In *Proceedings of the AAAI Spring Symposium on Intelligent Narrative Technologies II*.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *In Proceedings of COLING/ACL*, pages 86–90.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of the Association for Computational Linguistics*.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Association for Computational Linguistics*.

Yejin Choi, Eric Breck, and Claire Cardie. 2006. Joint extraction of entities and relations for opinion recognition. In *EMNLP '06: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 431–439, Morristown, NJ, USA. Association for Computational Linguistics.

David Elson and Kathleen McKeown. 2009. Extending and evaluating a platform for story understanding. In *Proceedings of the AAAI 2009 Spring Symposium on Intelligent Narrative Technologies II*.

Toshiaki Fujiki, Hidetsugu Nanba, and Manabu Okumura. 2003. Automatic acquisition of script knowledge from a text collection. In *Proceedings of the European Association for Computational Linguistics*.

Vasileios Hatzivassiloglou and Kathy McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 174–181, Madrid, Spain.

Paul J. Hopper and Sandra A. Thompson. 1980. Transitivity in grammar and discourse. *Language*, 56:251299.

Niels Kasch and Tim Oates. 2010. Mining script-like structures from the web. In *NAACL-10 Workshop on Formalisms and Methodology for Learning by Reading (FAM-LbR)*.

S. Kim and E. Hovy. 2006. Extracting Opinions, Opinion Holders, and Topics Expressed in Online News Media Text. In *Proceedings of ACL/COLING Workshop on Sentiment and Subjectivity in Text*.

W. Lehnert, J. Black, and B. Reiser. 1981. Summarizing Narratives. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*.

W. G. Lehnert. 1981. Plot Units and Narrative Summarization. *Cognitive Science*, 5(4):293–331.

G. Miller. 1990. Wordnet: An On-line Lexical Database. *International Journal of Lexicography*, 3(4).

Raymond Mooney and Gerald DeJong. 1985. Learning Schemata for Natural Language Processing. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 681–687.

E. Riloff and W. Phillips. 2004. An Introduction to the Sundance and AutoSlog Systems. Technical Report UUCS-04-015, School of Computing, University of Utah.

E. Riloff, J. Wiebe, and T. Wilson. 2003. Learning Subjective Nouns using Extraction Pattern Bootstrapping. In *Proceedings of the Seventh Conference on Natural Language Learning (CoNLL-2003)*, pages 25–32.

Roger C. Schank and Robert P. Abelson. 1977. *Scripts, plans, goals and understanding*. Lawrence Erlbaum.

V. Stoyanov and C. Cardie. 2008. Topic Identification for Fine-Grained Opinion Analysis. In *Conference on Computational Linguistics (COLING 2008)*.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Extracting semantic orientations of words using spin model. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*.

M. Thelen and E. Riloff. 2002. A Bootstrapping Method for Learning Semantic Lexicons Using Extraction Pa ttern Contexts. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 214–221.

A. Wierzbicka. 1987. *English speech act verbs: a semantic dictionary*. Academic Press, Sydney, Orlando.

T. Wilson, P. Hoffmann, S. Somasundaran, J. Kessler, J. Wiebe, Y. Choi, C. Cardie, E. Riloff, and S. Patwardhan. 2005a. OpinionFinder: A system for subjectivity analysis. In *Proceedings of HLT/EMNLP 2005 Interactive Demonstrations*.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005b. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 347–354. Association for Computational Linguistics.

Handling Noisy Queries In Cross Language FAQ Retrieval

Danish Contractor Govind Kothari Tanveer A. Faruquie
L. Venkata Subramaniam Sumit Negi
IBM Research India
Vasant Kunj, Institutional Area
New Delhi, India
{dcontrac,govkotha,ftanveer,lvsubram,sumitneg}@in.ibm.com

Abstract

Recent times have seen a tremendous growth in mobile based data services that allow people to use Short Message Service (SMS) to access these data services. In a multilingual society it is essential that data services that were developed for a specific language be made accessible through other local languages also. In this paper, we present a service that allows a user to query a Frequently-Asked-Questions (FAQ) database built in a local language (Hindi) using Noisy SMS English queries. The inherent noise in the SMS queries, along with the language mismatch makes this a challenging problem. We handle these two problems by formulating the query similarity over FAQ questions as a combinatorial search problem where the search space consists of combinations of dictionary variations of the noisy query and its top-N translations. We demonstrate the effectiveness of our approach on a real-life dataset.

1 Introduction

There has been a tremendous growth in the number of new mobile subscribers in the recent past. Most of these new subscribers are from developing countries where mobile is the primary information device. Even for users familiar with computers and the internet, the mobile provides unmatched portability. This has encouraged the proliferation of information services built around SMS technology. Several applications, traditionally available on Internet, are now being made available on mobile devices using SMS. Examples include SMS short code services. Short codes are numbers where a short message in a predesignated format can be sent to get specific information. For example, to get the closing stock price of a particular share, the user has to send a message IBMSTOCKPR. Other examples are search (Schusteritsch et al., 2005), access to Yellow Page services (Kopparapu et al., 2007), Email [1], Blog [2] , FAQ retrieval [3] etc. The SMS-based FAQ retrieval services use human experts to answer SMS questions.

Recent studies have shown that instant messaging is emerging as the preferred mode of communication after speech and email.[4] Millions of users of instant messaging (IM) services and short message service (SMS) generate electronic content in a dialect that does not adhere to conventional grammar, punctuation and spelling standards. Words are intentionally compressed by non-standard spellings, abbreviations and phonetic transliteration are used. Typical question answering systems are built for use with languages which are free from such errors. It is difficult to build an automated question answering system around SMS technology. This is true even for questions whose answers are well documented like in a Frequently-Asked-Questions (FAQ) database. Unlike other automatic question answering systems that focus on searching answers from a given text collection, Q&A archive (Xue et al., 2008) or the Web (Jijkoun et al., 2005), in a FAQ database the questions and answers are already pro-

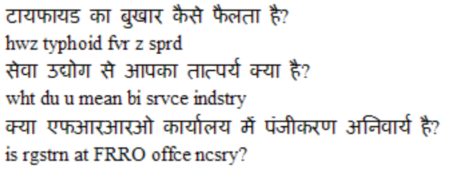
[1] http://www.sms2email.com/

[2] http://www.letmeparty.com/

[3] http://www.chacha.com/

[4] http://www.whyconverge.com/

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 87–96,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics



टायफायड का बुखार कैसे फैलता है?
hwz typhoid fvr z sprd
सेवा उद्योग से आपका तात्पर्य क्या है?
wht du u mean bi srvce indstry
क्या एफआरआरओ कार्यालय में पंजीकरण अनिवार्य है?
is rgstrn at FRRO offce ncsry?

Figure 1: Sample SMS queries with Hindi FAQs

vided by an expert. The main task is then to identify the best matching question to retrieve the relevant answer (Sneiders, 1999) (Song et al., 2007). The high level of noise in SMS queries makes this a difficult problem (Kothari et al., 2009). In a multilingual setting this problem is even more formidable. Natural language FAQ services built for users in one language cannot be accessed in another language. In this paper we present a FAQ-based question answering system over a SMS interface that solves this problem for two languages. We allow the FAQ to be in one language and the SMS query to be in another.

Multi-lingual question answering and information retrieval has been studied in the past (Sekine and Grishman, 2003)(Cimiano et al., 2009). Such systems resort to machine translation so that the search can be performed over a single language space. In the two language setting, it involves building a machine translation system engine and using it such that the question answering system built for a single language can be used.

Typical statistical machine translation systems use large parallel corpora to learn the translation probabilities (Brown et al., 2007). Traditionally such corpora have consisted of news articles and other well written articles. Since the translation systems are not trained on SMS language they perform very poorly when translating noisy SMS language. Parallel corpora comprising noisy sentences in one language and clean sentences in another language are not available and it would be hard to build such large parallel corpora to train a machine translation system. There exists some work to remove noise from SMS (Choudhury et al., 2007) (Byun et al., 2008) (Aw et al., 2006) (Neef et al., 2007) (Kobus et al., 2008). However, all of these techniques require an aligned corpus of SMS and conventional language for training. Such data is extremely hard to create. Unsupervised techniques require huge amounts of SMS data to learn mappings of non-standard words to their corresponding conventional form (Acharyya et al., 2009).

Removal of noise from SMS without the use of parallel data has been studied but the methods used are highly dependent on the language model and the degree of noise present in the SMS (Contractor et al., 2010). These systems are not very effective if the SMSes contain grammatical errors (or the system would require large amounts of training data in the language model to be able to deal with all possible types of noise) in addition to misspellings etc. Thus, the translation of a cleaned SMS, into a second language, will not be very accurate and it would not give good results if such a translated SMS is used to query an FAQ collection.

Token based noise-correction techniques (such as those using edit-distance, LCS etc) cannot be directly applied to handle the noise present in the SMS query. These noise-correction methods return a list of candidate terms for a given noisy token (E.g. ’gud’ $->$ ’god’,’good’,’guide’) . Considering all these candidate terms and their corresponding translations drastically increase the search space for any multi-lingual IR system. Also , naively replacing the noisy token in the SMS query with the top matching candidate term gives poor performance as shown by our experiments. Our algorithm handles these and related issues in an efficient manner.

In this paper we address the challenges arising when building a cross language FAQ-based question answering system over an SMS interface. Our method handles noisy representation of questions in a source language to retrieve answers across target languages. The proposed method does not require hand corrected data or an aligned corpus for explicit SMS normalization to mitigate the effects of noise. It also works well with grammatical noise. To the best of our knowledge we are the first to address issues in noisy SMS based cross-language FAQ retrieval. We propose an efficient algorithm that can handle noise in the form of lexical and semantic corruptions in the source language.

2 Problem formulation

Consider an input SMS S^e in a source language e. We view S^e as a sequence of n tokens $S^e = s_1, s_2, \ldots, s_n$. As explained in the introduction, the input is bound to have misspellings and other lexical and semantic distortions. Also let Q^h denote the set of questions in the FAQ corpus of a target language h. Each question $Q^h \in \mathcal{Q}^h$ is also viewed as a sequence of tokens. We want to find the question $\hat{Q^h}$ from the corpus $\mathcal{Q}^h$ that best matches the SMS S^e.

The matching is assisted by a source dictionary $\mathcal{D}^e$ consisting of clean terms in e constructed from a general English dictionary and a domain dictionary of target language $\mathcal{D}^h$ built from all the terms appearing in $\mathcal{Q}^h$. For a token s_i in the SMS input, term t_e in dictionary $\mathcal{D}^e$ and term t_h in dictionary $\mathcal{D}^h$ we define a *cross-lingual similarity measure* $\alpha(t_h, t_e, s_i)$ that measures the extent to which term s_i matches t_h using the clean term t_e. We consider t_h a *cross lingual variant* of s_i if for any t_e the cross language similarity measure $\alpha(t_h, t_e, s_i) > \epsilon$. We denote this as $t_h \sim s_i$.

We define a weight function $\omega(t_h, t_e, s_i)$ using the *cross lingual similarity measure* and the *inverse document frequency* (idf) of t_h in the target language FAQ corpus. We also define a scoring function to assign a score to each question in the corpus $\mathcal{Q}^h$ using the weight function. Consider a question $Q^h \in \mathcal{Q}^h$. For each token s_i, the scoring function chooses the term from Q^h having the maximum weight using possible clean representations of s_i; then the weight of the n chosen terms are summed up to get the score. The score measures how closely the question in FAQ matches the noisy SMS string S^e using the composite weights of individual tokens.

$$\text{Score}(Q^h) = \sum_{i=1}^{n} \max_{t_h \in Q^h, t_e \in \mathcal{D}^e \text{ \& } t_h \sim s_i} \omega(t_h, t_e, s_i)$$

Our goal is to efficiently find the question $\hat{Q^h}$ having the maximum score.

3 Noise removal from queries

In order to process the noisy SMS input we first have to map noisy tokens in S^e to the possible correct lexical representations. We use a similarity measure to map the noisy tokens to their clean lexical representations.

3.1 Similarity Measure

For a term $t_e \in \mathcal{D}^e$ and token s_i of the SMS input S^e, the similarity measure $\gamma(t_e, s_i)$ between them is

$$\gamma(t_e, s_i) = \begin{cases} \frac{LCSRatio(t_e,s_i)}{EditDistance_{SMS}(t_e,s_i)} & \textit{if } t_e \textit{ and } s_i \textit{ share same starting character *} \\ 0 & \textit{otherwise} \end{cases} \tag{1}$$

Where $LCSRatio(t_e, s_i) = \frac{length(LCS(t_e,s_i))}{length(t_e)}$ and $LCS(t_e, s_i)$ is the *Longest common subsequence* between t_e and s_i.
** The intuition behind this measure is that people typically type the first few characters of a word in an SMS correctly. This way we limit the possible variants for a particular noisy token*

The *Longest Common Subsequence Ratio* (LCSRatio) (Melamed et al., 1999) of two strings is the ratio of the length of their LCS and the length of the longer string. Since in the SMS scenario, the dictionary term will always be longer than the SMS token, the denominator of LCSRatio is taken as the length of the dictionary term.

The $EditDistance_{SMS}$ (Figure 2) compares the Consonant Skeletons (Prochasson et al., 2007) of the dictionary term and the SMS token. If the Levenshtein distance between consonant skeletons is small then $\gamma(t_e, s_i)$ will be high. The intuition behind using $EditDistance_{SMS}$ can be explained through an example. Consider an SMS token "gud" whose most likely correct form is "good". The longest common subsequence for "good" and "guided" with "gud" is "gd". Hence the two dictionary terms "good" and "guided" have the same LCSRatio of 0.5 w.r.t "gud", but the $EditDistance_{SMS}$ of "good" is 1 which is less than that of "guided", which has $EditDistance_{SMS}$ of 2 w.r.t "gud". As a result the similarity measure between "gud" and "good" will be higher than that of "gud" and "guided". Higher the *LCSRatio* and lower the $EditDistance_{SMS}$, higher will be the similarity measure. Hence, for a given SMS token "byk", the similarity measure of word "bike" is higher than that of "break".

4 Cross lingual similarity

Once we have potential candidates which are the likely disambiguated representations of the noisy

```
Procedure $EditDistance_{SMS}(t_e, s_i)$
Begin
  return $LevenshteinDistance(CS(s_i), CS(t_e)) + 1$
End

Procedure CS (t): // Consonant Skeleton Generation
Begin
  Step 1. remove consecutive repeated characters in $t$
    // ($fall \rightarrow fal$)
  Step 2. remove all vowels in $t$
    //($painting \rightarrow pntng, threat \rightarrow thrt$)
  return $t$
End
```

Figure 2: $EditDistance_{SMS}$

term, we map these candidates to appropriate terms in the target language. We use a statistical dictionary to achieve this cross lingual mapping.

4.1 Statistical Dictionary

In order to build a statistical dictionary we use the statistical translation model proposed in (Brown et al., 2007). Under IBM model 2 the translation probability of source language sentence $\bar{e} = \{t_e^1, \ldots, t_e^j, \ldots, t_e^m\}$ and a target language sentence $\bar{h} = \{t_h^1, \ldots, t_h^i, \ldots, t_e^l\}$ is given by

$$Pr(\bar{h}|\bar{e}) = \phi(l|m) \prod_{i=1}^{l} \sum_{j=0}^{m} \tau(t_h^i|t_e^j) a(j|i, m, l). \tag{2}$$

Here the word translation model $\tau(t_h|t_e)$ gives the probability of translating the source term to target term and the alignment model $a(j|i, m, l)$ gives the probability of translating the source term at position i to a target position j. This model is learnt using an aligned parallel corpus.

Given a clean term t_e^i in source language we get all the corresponding terms $\mathcal{T} = \{t_h^1, \ldots, t_h^k, \ldots\}$ from the target language such that word translation probability $\tau(t_h^k|t_e^i) > \varepsilon$. We rank these terms according to the probability given by the word translation model $\tau(t_h|t_e)$ and consider only those target terms that are part of domain dictionary i.e. $t_h^k \in \mathcal{D}^h$.

4.2 Cross lingual similarity measure

For each term s_i in SMS input query, we find all the clean terms t_e in source dictionary $\mathcal{D}^e$ for which similarity measure $\gamma(t_e, s_i) > \phi$. For each of these term t_e, we find the cross lingual similar terms $\mathcal{T}_{t_e}$ using the word translation model. We compute the cross lingual similarity measure between these terms as

$$\alpha(s_i, t_e, t_h) = \gamma(t_e, s_i).\tau(t_h, t_e) \tag{3}$$

The measure selects those terms in target language that have high probability of being translated from a noisy term through one or more valid clean terms.

4.3 Cross lingual similarity weight

We combine the idf and the cross lingual similarity measure to define the cross lingual weight function $\omega(t_h, t_e, s_i)$ as

$$\omega(t_h, t_e, s_i) = \alpha(t_h, t_e, s_i).idf(t_h) \tag{4}$$

By using idf we give preference to terms that are highly discriminative. This is necessary because queries are distinguished from each other using informative words. For example for a given noisy token "bck" if a word translation model produces a translation output "wapas" (as in came back) or "peet" or "qamar" (as in back pain) then idf will weigh "peet" more as it is relatively more discriminative compared to "wapas" which is used frequently.

5 Pruning and matching

In this section we describe our search algorithm and the preprocessing needed to find the best question $\hat{Q}^h$ for a given SMS query.

5.1 Indexing

Our algorithm operates at a token level and its corresponding cross lingual variants. It is therefore necessary to be able to retrieve all questions $Q_{t_h}^h$ that contain a given target language term t_h. To do this efficiently we index the questions in FAQ corpus using *Lucene*[5]. Each question in FAQ is treated as a document. It is tokenized using whitespace as delimiter before indexing.

[5] http://lucene.apache.org/java/docs/

The cross lingual similarity weight calculation requires the *idf* for a given term t_h. We query on this index to determine the number of documents f that contain t_h. The *idf* of each term in $\mathcal{D}^h$ is precomputed and stored in a hashtable with t_h as the key. The cross lingual similarity measure calculation requires the word translation probability for a given term t_e. For every t_e in dictionary D^e, we store $\mathcal{T}_{t_e}$ in a hashmap that contains a list of terms in the target language along with their statistically determined translation probability $\tau(t_h|t_e) > \varepsilon$, where $t_h \in \mathcal{D}^h$.

Since the query and the FAQs use terms from different languages, the computation of IDF becomes a challenge (Pirkola, 1998) (Oard et al., 2007). Prior work uses a bilingual dictionary for translations for calculating the IDF. We on the other hand rely on a statistical dictionary that has translation probabilities. Applying the method suggested in the prior work on a statistical dictionary leads to errors as the translations may themselves be inaccurate.

We therefore calculate IDFs for target language term (translation) and use it in the weight measure calculation. The method suggested by Oard et al (Oard et al., 2007) is more useful in retrieval tasks for multiple documents, while in our case we need to retrieve a specific document (FAQ).

5.2 List Creation

Given an SMS input string S^e, we tokenize it on white space and replace any occurrence of digits to their string based form (e.g. 4get, 2day) to get a series of n tokens $s_1, s_2, \ldots, s_n$. A list L_i^e is created for each token s_i using terms in the monolingual dictionary D^e. The list for a single character SMS token is set to *null* as it is most likely to be a stop word. A term t_e from D^e is included in L_i^e if it satisfies the threshold condition

$$\gamma(t_e, s_i) > \phi \tag{5}$$

The threshold value ϕ is determined experimentally. For every $t_e \in L_i^e$ we retrieve $\mathcal{T}_{t_e}$ and then retrieve the *idf* scores for every $t_h \in \mathcal{T}_{t_e}$. Using the word translation probabilities and the idf score we compute the cross lingual similarity weight to create a new list L_i^h. A term t_h is included in the list only if

$$\tau(t_h|t_e) > 0.1 \tag{6}$$

This probability cut-off is used to prevent poor quality translations from being included in the list.

If more than one term t_e has the same translation t_h, then t_h can occur more than once in a given list. If this happens, then we remove repetitive occurrences of t_h and assign it a weight equal to the maximum weight amongst all occurrences in the list, multiplied by the number of times it occurs. The terms t_h in L_i^h are sorted in decreasing order of their similarity weights. Henceforth, the term “list” implies a sorted list.

For example given a SMS query “hw mch ds it cst to stdy in india” as shown in Fig. 3, for each token we create a list of possible correct dictionary words by dictionary look up. Thus for token “cst” we get dictionary words lik “cost, cast, case, close”. For each dictionary word we get a set of possible words in Hindi by looking at statistical translation table. Finally we merged the list obtained to get single list of Hindi words. The final list is ranked according to their similarity weights.

5.3 Search algorithm

Given S^e containing n tokens, we create n sorted lists $L_1^h, L_2^h, \ldots, L_n^h$ containing terms from the domain dictionary and sorted according to their cross lingual weights as explained in the previous section. A naive approach would be to query the index using each term appearing in all L_i^h to build a *Collection set* $\mathcal{C}$ of questions. The best matching question $\hat{Q}^h$ will be contained in this collection. We compute the score of each question in $\mathcal{C}$ using $Score(Q)$ and the question with highest score is treated as $\hat{Q}^h$. However the naive approach suffers from high runtime cost.

Inspired by the Threshold Algorithm (Fagin et al., 2001) we propose using a pruning algorithm that maintains a much smaller *candidate set* $\mathcal{C}$ of questions that can potentially contain the maximum scoring question. The algorithm is shown in Figure 4. The algorithm works in an iterative manner. In each iteration, it picks the term that has maximum weight among all the terms appearing in the lists $L_1^h, L_2^h, \ldots, L_n^h$. As the lists are sorted in the descending order of the weights, this amounts to picking the maximum weight term amongst the first terms of the n lists. The chosen term t_h is queried to find the set Q_{t_h}. The set Q_{t_h} is added to the candi-

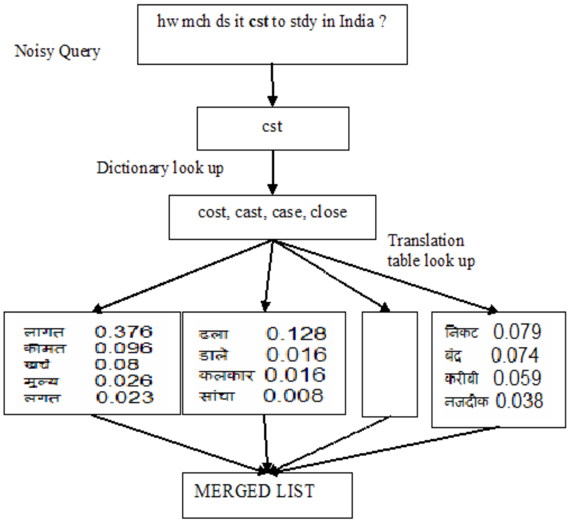


Figure 3: List creation

date set $\mathcal{C}$. For each question $Q \in Q_{t_h}$, we compute its score $Score(Q)$ and keep it along with Q. After this the chosen term t_h is removed from the list and the next iteration is carried out. We stop the iterative process when a thresholding condition is met and focus only on the questions in the candidate set $\mathcal{C}$. The thresholding condition guarantees that the candidate set $\mathcal{C}$ contains the maximum scoring question $\hat{Q}^h$. Next we develop this thresholding condition.

Let us consider the end of an iteration. Suppose Q is a question not included in $\mathcal{C}$. At best, Q will include the current top-most tokens $L_1^h[1], L_2^h[1], \ldots, L_n^h[1]$ from every list. Thus, the upper bound UB on the score of Q is

$$\text{Score}(Q) \leq \sum_{i=0}^{n} \omega(L_i^h[1]).$$

Let Q^* be the question in $\mathcal{C}$ having the maximum score. Notice that if $Q^* \geq$ UB, then it is guaranteed that any question not included in the candidate set $\mathcal{C}$ cannot be the maximum scoring question. Thus, the condition "$Q^* \geq$ UB" serves as the termination criterion. At the end of each iteration, we check if the termination condition is satisfied and if so, we can stop the iterative process. Then, we simply pick the question in $\mathcal{C}$ having the maximum score and return it.

```
Procedure Search Algorithm
Input: SMS $S = s_1, s_2, \ldots, s_n$
Output: Maximum scoring question $\hat{Q}^h$.
Begin
  $\forall s_i$, construct $L_i^e$ for which $\gamma(s_i, t_e) > \epsilon$
    // $L_i$ lists variants of $s_i$
  Construct lists $L_1^h, L_2^h, \ldots, L_n^h$ //(see Section 5.2).
    // $L_i^h$ lists cross lingual variants of $s_i$ in decreasing
    //order of weight.
  Candidate list $\mathcal{C} = \emptyset$.
  repeat
    $j^* = \text{argmax}_i \omega(L_i^h[1])$
    $t_h^* = L_{j^*}^h[1]$
      // $t_h^*$ is the term having maximum weight among
      // all terms appearing in the $n$ lists.
    Delete $t_h^*$ from the list $L_{j^*}^h$.
    Retrieve $Q_{t_h^*}$ using the index
      // $Q_{t_h^*}$: the set of all questions in $\mathcal{Q}^h$
      //having the term $t_h^*$
    For each $Q \in Q_{t_h^*}$
      Compute Score($Q$) and
      add $Q$ with its score into $\mathcal{C}$
    UB = $\sum_{i=1}^{n} \omega(L_i^h[1])$
    $\widehat{Q} = \text{argmax}_{Q \in \mathcal{C}} \text{Score}(Q)$.
    if Score($\widehat{Q}$) $\geq$ UB, then
        // Termination condition satisfied
      Output $\widehat{Q}$ and exit.
  forever
End
```

Figure 4: Search Algorithm with Pruning

6 Experiments

To evaluate our system we used noisy English SMS queries to query a collection of $10,000$ Hindi FAQs. These FAQs were collected from websites of various government organizations and other online resources. These FAQs are related to railway reservation, railway enquiry, passport application and health related issues. For our experiments we asked 6 human evaluators, proficient in both English and Hindi, to create English SMS queries based on the general topics that our FAQ collection dealt with. We found 60 SMS queries created by the evaluators, had answers in our FAQ collection and we designated these as the in-domain queries. To measure the effectiveness of our system in handling out of domain queries we used a total of 380 SMSes part of which were taken from the NUS corpus (How et al.,

whch metro statn z nr pragati maidan ?
dus metro goes frm airpot 2 new delhi rlway statn?
is dere any special metro pas 4 delhi uni students?
whn is d last train of delhi metro?
whr r d auto stands N delhi?

Figure 5: Sample SMS queries

2005) and the rest from the "out-of-domain" queries created by the human evaluators. Thus the total SMS query data size was 440. Fig 5 shows some of the sample queries.

Our objective was to retrieve the correct Hindi FAQ response given a noisy English SMS query. A given English SMS query was matched against the list of indexed FAQs and the best matching FAQ was returned by the Pruning Algorithm described in Section 5. A score of 1 was assigned if the retrieved answer was indeed the response to the posed SMS query else we assigned a score of 0. In case of out of domain queries a score of 1 was assigned if the output was NULL else we assigned a score of 0.

6.1 Translation System

We used the Moses toolkit (Koehn et al., 2007) to build an English-Hindi statistical machine translation system. The system was trained on a collection of $150,000$ English and Hindi parallel sentences sourced from a publishing house. The $150,000$ sentences were on a varied range of subjects such as news, literature, history etc. Apart from this the training data also contained an aligned parallel corpus of English and Hindi FAQs. The FAQs were collected from government websites on topics such as health, education, travel services etc.

Since an MT system trained solely on a collection of sentences would not be very accurate in translating questions, we trained the system on an English-Hindi parallel question corpus. As it was difficult to find a large collection of parallel text consisting of questions, we created a small collection of parallel questions using 240 FAQs and multiplied them to create a parallel corpus of $50,000$ sentences. This set was added to the training data and this helped familiarize the language model and phrase tables used by the MT systems to questions. Thus in total the MT system was trained on a corpus of $200,000$ sentences.

Experiment 1 and 2 form the baseline against which we evaluated our system. For our experiments the lexical translation probabilities generated by Moses toolkit were used to build the word translation model. In Experiment 1 the threshold ϕ described in Equation 5 is set to 1. In Experiment 2 and 3 this is set to 0.5. The Hindi FAQ collection was indexed using Lucene and a domain dictionary D^h was created from the Hindi words in the FAQ collection.

6.2 System Evaluation

We perform three sets of experiments to show how each stage of the algorithm contributes in improving the overall results.

6.2.1 Experiment 1

For Experiment 1 the threshold ϕ in Equation 5 is set to 1 i.e. we consider only those tokens in the query which belong to the dictionary. This setup illustrates the case when no noise handling is done. The results are reported in Figure 6.

6.2.2 Experiment 2

For Experiment 2 the noisy SMS query was cleaned using the following approach. Given a noisy token in the SMS query it's similarity (Equation 1) with each word in the Dictionary is calculated. The noisy token is replaced with the Dictionary word with the maximum similarity score. This gives us a clean English query.

For each token in the cleaned English SMS query, we create a list of possible Hindi translations of the token using the statistical translation table. Each Hindi word was assigned a weight according to Equation 4. The Pruning algorithm in Section 5 was then applied to get the best matching FAQ.

6.2.3 Experiment 3

In this experiment, for each token in the noisy English SMS we obtain a list of possible English variations. For each English variation a corresponding set of Hindi words from the statistical translation table was obtained. Each Hindi word was assigned a weight according to Equation 4. As described in Section 5.2, all Hindi words obtained from English variations of a given SMS token are merged to create

	Experiment 1	Experiment 2	Experiment 3
MRR Score	0.41	0.68	0.83

Table 1: MRR Scores

	F1 Score
Expt 1 (Baseline 1)	0.23
Expt 2 (Baseline 2)	0.68
Expt 3 (Proposed Method)	0.72

Table 2: F1 Measure

a list of Hindi words sorted in terms of their weight. The Pruning algorithm as described in Section 5 was then applied to get the best matching FAQ.

We evaluated our system using two different criteria. We used MRR (Mean reciprocal rank) and the best matching accuracy. Mean reciprocal rank is used to evaluate a system by producing a list of possible responses to a query, ordered by probability of correctness. The reciprocal rank of a query response is the multiplicative inverse of the rank of the first correct answer. The mean reciprocal rank is the average of the reciprocal ranks of results for a sample of queries Q.

$$MRR = 1/|Q| \sum_{i=1}^{Q} 1/rank_i \tag{7}$$

Best match accuracy can be considered as a special case of MRR where the size of the ranked list is 1. As the SMS based FAQ retrieval system will be used via mobile phones where screen size is a major constraint it is crucial to have the correct result on the top. Hence in our settings the best match accuracy is a more relevant and stricter performance evaluation measure than MRR.

Table 1 compares the MRR scores for all three experiments. Our method reports the highest MRR of 0.83. Figure 6 shows the performance using the strict evaluation criterion of the top result returned being correct.

We also experimented with different values of the threshold for Score(Q) (Section 5.3). The ROC curve for various threshold is shown in Figure 7. The result for both in-domain and out-of-domain queries for the three experiments are shown in Figure 6 for Score(Q) = 8. The F1 Score for experiments 1, 2 and 3 are shown in Table 2.

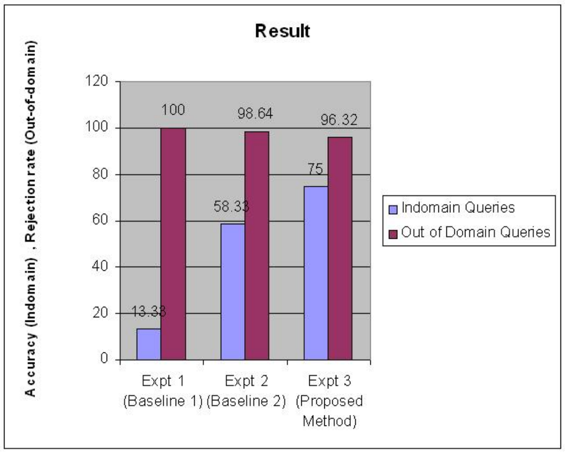


Figure 6: Comparison of results

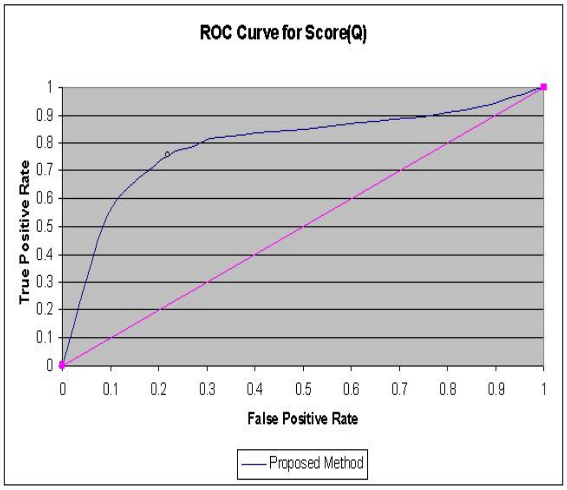


Figure 7: ROC Curve for $Score(Q)$

6.3 Measuring noise level in SMS queries

In order to quantify the level of noise in the collected SMS data, we built a character-level language model(LM) using the questions in the FAQ data-set (vocabulary size is 70) and computed the perplexity of the language model on the noisy and the cleaned SMS test-set. The perplexity of the LM on a corpus gives an indication of the average number of bits needed per n-gram to encode the corpus. Noise re-

		Cleaned SMS	Noisy SMS
English FAQ collection	bigram	16.64	55.19
	trigram	9.75	69.41

Table 3: Perplexity for Cleaned and Noisy SMS

sults in the introduction of many previously unseen n-grams in the corpus. Higher number of bits are needed to encode these improbable n-grams which results in increased perplexity. From Table 3 we can see the difference in perplexity for noisy and clean SMS data for the English FAQ data-set. Large perplexity values for the SMS dataset indicates a high level of noise.

For each noisy SMS query e.g. “hw 2 prvnt typhd” we manually created a clean SMS query “how to prevent typhoid”. A character level language model using the questions in the clean English FAQ dataset was created to quantify the level of noise in our SMS dataset. We computed the perplexity of the language model on clean and noisy SMS queries.

7 Conclusion

There has been a tremendous increase in information access services using SMS based interfaces. However, these services are limited to a single language and fail to scale for multilingual QA needs. The ability to query a FAQ database in a language other than the one for which it was developed is of great practical significance in multilingual societies. Automatic cross-lingual QA over SMS is challenging because of inherent noise in the query and the lack of cross language resources for noisy processing. In this paper we present a cross-language FAQ retrieval system that handles the inherent noise in source language to retrieve FAQs in a target language. Our system does not require an end-to-end machine translation system and can be implemented using a simple dictionary which can be static or constructed statistically using a moderate sized parallel corpus. This side steps the problem of building full fledged translation systems but still enabling the system to be scaled across multiple languages quickly. We present an efficient algorithm to search and match the best question in the large FAQ corpus of target language for a noisy input question. We have demonstrated the effectiveness of our approach on a real life FAQ corpus.

References

Sreangsu Acharyya, Sumit Negi, L Venkata Subramaniam, Shourya Roy. 2009. Language independent unsupervised learning of short message service dialect. *International Journal on Document Analysis and Recognition*, pp. 175-184.

Aiti Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A phrase-based statistical model for SMS text normalization. *In Proceedings of COLING-ACL*, pp. 33-40.

Peter F. Brown, Vincent J.Della Pietra, Stephen A. Della Pietra, Robert. L. Mercer 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation *Computational Linguistics*, pp. 263-311.

Jeunghyun Byun, Seung-Wook Lee, Young-In Song, Hae-Chang Rim. 2008. Two Phase Model for SMS Text Messages Refinement. *AAAI Workshop on Enhanced Messaging*.

Monojit Choudhury, Rahul Saraf, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar, Anupam Basu. 2007. Investigation and modeling of the structure of texting language. *International Journal on Document Analysis and Recognition*, pp. 157-174.

Philipp Cimiano, Antje Schultz, Sergej Sizov, Philipp Sorg, Steffen Staab. 2009. Explicit versus latent concept models for cross-language information retrieval. *In Proceeding of IJCAI*, pp. 1513-1518.

Danish Contractor, Tanveer A. Faruquie, L. Venkata Subramaniam. 2010. Unsupervised cleansing of noisy text. *In Proceeding of COLING 2010: Posters*, pp. 189-196.

R. Fagin, A. Lotem, and M. Naor. 2001. Optimal aggregation algorithms for middleware. *In Proceedings of the 20th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 102-113.

Yijue How and Min-Yen Kan. 2005. Optimizing predictive text entry for short message service on mobile phones. *In M. J. Smith and G. Salvendy (Eds.) Proc. of Human Computer Interfaces International,Lawrence Erlbaum Associates*

Valentin Jijkoun and Maarten de Rijke. 2005. Retrieving answers from frequently asked questions pages on the web. *In Proceedings of the Tenth ACM Conference on Information and Knowledge Management,CIKM*, pp. 76-83.

Catherine Kobus, Francois Yvon and Grraldine Damnati. 2008. Normalizing SMS: Are two metaphors better than one? *In Proceedings of COLING*, pp. 441-448.

Philipp Koehn, Hieu Hoang, Alexandra Birch Mayne, Christopher Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, Evan Herbst 2007. Moses: Open source toolkit for statistical machine translation. *Annual Meeting of the Association for Computation Linguistics (ACL), Demonstration Session* .

Sunil Kumar Kopparapu, Akhilesh Srivastava and Arun Pande. 2007. SMS based Natural Language Interface to Yellow Pages Directory. *In Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology*, pp. 558-563 .

Govind Kothari, Sumit Negi, Tanveer Faruquie, Venkat Chakravarthy and L V Subramaniam 2009. SMS based Interface for FAQ Retrieval. *Annual Meeting of the Association for Computation Linguistics (ACL).*

I. D. Melamed. 1999. Bitext maps and alignment via pattern recognition. *Computational Linguistics*, pp. 107-130.

Guimier de Neef, Emilie, Arnaud Debeurme, and Jungyeul Park. 2007. TILT correcteur de SMS : Evaluation et bilan quantitatif. *In Actes de TALN*, pp. 123-132.

Douglas W. Oard, Funda Ertunc. 2002. Translation-Based Indexing for Cross-Language Retrieval *In Proceedings of the ECIR*, pp. 324-333.

A. Pirkola 1998. The Effects of Query Structure and Dictionary Setups in Dictionary-Based Cross-Language Information Retrieval *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* , pp. 55-63.

E. Prochasson, C. Viard-Gaudin, and E. Morin. 2007. Language models for handwritten short message services. *In Proceedings of the 9th International Conference on Document Analysis and Recognition*, pp. 83-87.

Rudy Schusteritsch, Shailendra Rao, Kerry Rodden. 2005. Mobile Search with Text Messages: Designing the User Experience for Google SMS. *In Proceedings of ACM SIGCHI*, pp. 1777-1780.

Satoshi Sekine, Ralph Grishman. 2003. Hindi-English cross-lingual question-answering system. *ACM Transactions on Asian Language Information Processing*, pp. 181-192.

E. Sneiders. 1999. Automated FAQ Answering: Continued Experience with Shallow Language Understanding Question Answering Systems. *Papers from the 1999 AAAI Fall Symposium*. Technical Report FS-99-02, AAAI Press, pp. 97-107.

W. Song, M. Feng, N. Gu, and L. Wenyin. 2007. Question similarity calculation for FAQ answering. *In Proceeding of SKG 07*, pp. 298-301.

X. Xue, J. Jeon, and W.B Croft. 2008. Retrieval Models for Question and Answer Archives. *In Proceedings of SIGIR*, pp. 475-482.

Learning the Relative Usefulness of Questions in Community QA

Razvan Bunescu
School of EECS
Ohio University
Athens, OH 43201, USA
bunescu@ohio.edu

Yunfeng Huang
School of EECS
Ohio University
Athens, OH 43201, USA
yh324906@ohio.edu

Abstract

We present a machine learning approach for the task of ranking previously answered questions in a question repository with respect to their relevance to a new, unanswered reference question. The ranking model is trained on a collection of question groups manually annotated with a partial order relation reflecting the relative utility of questions inside each group. Based on a set of meaning and structure aware features, the new ranking model is able to substantially outperform more straightforward, unsupervised similarity measures.

1 Introduction

Open domain Question Answering (QA) is one of the most complex and challenging tasks in natural language processing. In general, a question answering system may need to integrate knowledge coming from a wide variety of linguistic processing tasks such as syntactic parsing, semantic role labeling, named entity recognition, and anaphora resolution (Prager, 2006). State of the art implementations of these linguistic analysis tasks are still limited in their performance, with errors that compound and propagate into the final performance of the QA system (Moldovan et al., 2002). Consequently, the performance of open domain QA systems has yet to arrive at a level at which it would become a feasible alternative to the current paradigms for information access based on keyword searches.

Recently, community-driven QA sites such as Yahoo! Answers and WikiAnswers [1] have established a new approach to question answering that shifts the inherent complexity of open domain QA from the computer system to volunteer contributors. The computer is no longer required to perform a deep linguistic analysis of questions and generate corresponding answers, and instead acts as a mediator between users submitting questions and volunteers providing the answers.

An important objective in community QA is to minimize the time elapsed between the submission of questions by users and the subsequent posting of answers by volunteer contributors. One useful strategy for minimizing the response latency is to search the QA repository for similar questions that have already been answered, and provide the corresponding ranked list of answers, if such a question is found. The success of this approach depends on the definition and implementation of the question-to-question similarity function. In the simplest solution, the system searches for previously answered questions based on exact string matching with the reference question. Alternatively, sites such as WikiAnswers allow the users to mark questions they think are rephrasings ("alternate wordings", or paraphrases) of existing questions. These question clusters are then taken into account when performing exact string matching, therefore increasing the likelihood of finding previously answered questions that are semantically equivalent to the reference question.

In order to lessen the amount of work required from the contributors, an alternative approach is to build a system that automatically finds rephrasings of questions, especially since question rephrasing

[1] answers.yahoo.com, wiki.answers.com

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 97–107,
MIT, Massachusetts, USA, 9-11 October 2010.

seems to be computationally less demanding than question answering. According to previous work in this domain, a question is considered a rephrasing of a reference question Q_0 if it uses an alternate wording to express an identical information need. For example, Q_0 and Q_1 below are rephrasings of each other, and consequently they are expected to have the same answer.

Q_0 What should I feed my turtle?

Q_1 What do I feed my pet turtle?

Paraphrasings of a new question cannot always be found in the community QA repository. We believe that computing a ranked list of existing questions that at least partially address the original information need could also be useful to the user, at least until other users volunteer to give an exact answer to the original, unanswered reference question. For example, in the absence of any additional information about the reference question Q_0, the expected answers to questions Q_2 and Q_3 below may be seen as partially overlapping in information content with the expected answer for the reference question Q_0. An answer to question Q_4, on the other hand, is less likely to benefit the user, even though it has a significant lexical overlap with the reference question.

Q_2 What kind of fish should I feed my turtle?

Q_3 What do you feed a turtle that is the size of a quarter?

Q_4 What kind of food should I feed a turtle dove?

In this paper, we propose a supervised learning approach to the question ranking problem, a generalization of the question paraphrasing problem in which questions are ranked in a partial order based on the relative information overlap between their expected answers and the expected answer of the reference question. Underlying the question ranking task is the expectation that the user who submits a reference question will find the answers of the highly ranked questions to be more useful than the answers associated with the lower ranked questions. For the reference question Q_0 above, the learned ranking model is expected to produce a partial order in which Q_1 is ranked higher than Q_2, Q_3 and Q_4, whereas Q_2 and Q_3 are ranked higher than Q_4.

2 Partially Ordered Datasets for Question Ranking

In order to enable the evaluation of question ranking approaches, we have previously created a dataset of 60 groups of questions (Bunescu and Huang, 2010b). Each group consists of a reference question (e.g. Q_0 above) that is associated with a partially ordered set of questions (e.g. Q_1 to Q_4 above). For each reference questions, its corresponding partially ordered set is created from questions in Yahoo! Answers and other online repositories that have a high cosine similarity with the reference question. Out of the 26 top categories in Yahoo! Answers, the 60 reference questions span a diverse set of categories. Figure 1 lists the 20 categories covered, where each category is shown with the number of corresponding reference questions between parentheses.

```
   Travel (10), Computers & Internet (6),
    Beauty & Style (5), Entertainment &
Music (5), Food & Drink (5), Health (5),
      Arts & Humanities (3), Cars &
Transportation (3), Consumer Electronics
 (3), Pets (3), Family & Relationships
    (2), Science & Mathematics (2),
 Education & Reference (1), Environment
 (1), Local Businesses (1), Pregnancy &
 Parenting (1), Society & Culture (1),
     Sports (1), Yahoo! Products (1)
```

Figure 1: The 20 categories represented in the dataset.

Inside each group, the questions are manually annotated with a partial order relation, according to their utility with respect to the reference question. We use the notation $\langle Q_i \succ Q_j | Q_r \rangle$ to encode the fact that question Q_i is *more useful than* question Q_j with respect to the reference question Q_r. Similarly, $\langle Q_i = Q_j \rangle$ will be used to express the fact that questions Q_i and Q_j are reformulations of each other (the reformulation relation is independent of the reference question). The partial ordering among the questions Q_0 to Q_4 above can therefore be expressed concisely as follows: $\langle Q_0 = Q_1 \rangle$, $\langle Q_1 \succ Q_2 | Q_0 \rangle$, $\langle Q_1 \succ Q_3 | Q_0 \rangle$, $\langle Q_2 \succ Q_4 | Q_0 \rangle$, $\langle Q_3 \succ Q_4 | Q_0 \rangle$. Note that we do not explicitly annotate the relation $\langle Q_1 \succ Q_4 | Q_0 \rangle$, since it can be inferred based on the transitivity of the *more useful than* relation: $\langle Q_1 \succ Q_2 | Q_0 \rangle \wedge \langle Q_2 \succ Q_4 | Q_0 \rangle \Rightarrow \langle Q_1 \succ Q_4 | Q_0 \rangle$.

REFERENCE QUESTION (Q_r)

Q_5 What's a nice summer camp to go to in Florida?

PARAPHRASING QUESTIONS ($\mathcal{P}$)

Q_6 What camps are good for a vacation during the summer in FL?
Q_7 What summer camps in FL do you recommend?

USEFUL QUESTIONS ($\mathcal{U}$)

Q_8 Does anyone know a good art summer camp to go to in FL?
Q_9 Are there any good artsy camps for girls in FL?
Q_{10} What are some summer camps for like singing in Florida?
Q_{11} What is a good cooking summer camp in FL?
Q_{12} Do you know of any summer camps in Tampa, FL?
Q_{13} What is a good summer camp in Sarasota FL for a 12 year old?
Q_{14} Can you please help me find a surfing summer camp for beginners in Treasure Coast, FL?
Q_{15} Are there any acting summer camps and/or workshops in the Orlando, FL area?
Q_{16} Does anyone know any volleyball camps in Miramar, FL?
Q_{17} Does anyone know about any cool science camps in Miami?
Q_{18} What's a good summer camp you've ever been to?

NEUTRAL QUESTIONS ($\mathcal{N}$)

Q_{19} What's a good summer camp in Canada?
Q_{20} What's the summer like in Florida?

Table 1: A question group.

Also note that no relation is specified between Q_2 and Q_3, and similarly no relation can be inferred between these two questions. This reflects our belief that, in the absence of any additional information regarding the user or the "turtle" referenced in Q_0, we cannot compare questions Q_2 and Q_3 in terms of their usefulness with respect to Q_0.

Table 1 shows another reference question Q_5 from our dataset, together with its annotated group of questions Q_6 to Q_{20}. In order to make the annotation process easier and reproducible, we have divided it into two levels of annotation. During the first annotation stage, each question group is partitioned manually into 3 subgroups of questions:

- $\mathcal{P}$ is the set of *paraphrasing* questions.
- $\mathcal{U}$ is the set of *useful* questions.
- $\mathcal{N}$ is the set of *neutral* questions.

A question is deemed useful if its expected answer may overlap in information content with the expected answer of the reference question. The expected answer of a neutral question, on the other hand, should be irrelevant with respect to the reference question. Let Q_r be the reference question, $Q_p \in \mathcal{P}$ a paraphrasing question, $Q_u \in \mathcal{U}$ a useful question, and $Q_n \in \mathcal{N}$ a neutral question. Then the following relations are assumed to hold among these questions:

1. $\langle Q_p \succ Q_u | Q_r \rangle$: a *paraphrasing* question is more useful than a *useful* question.
2. $\langle Q_u \succ Q_n | Q_r \rangle$: a *useful* question is more useful than a *neutral* question.

Note that as long as these relations hold between the 3 types of questions, the names of the subgroups and their definitions are irrelevant with respect to the implied set of *more useful than* relations, since only the implied ternary relations will be used for training and evaluating question ranking approaches. We also assume that, by transitivity, the following ternary relations also hold: $\langle Q_p \succ Q_n | Q_r \rangle$, i.e. a *paraphrasing* question is more useful than a *neutral* question. Furthermore, if $Q_{p_1}, Q_{p_2} \in \mathcal{P}$ are two paraphrasing questions, this implies $\langle Q_{p_1} = Q_{p_2} | Q_r \rangle$.

For the vast majority of questions, the first annotation stage is straightforward and non-controversial. In the second annotation stage, we perform a finer annotation of relations between questions in the middle group $\mathcal{U}$. Table 1 shows two such relations (using indentation): $\langle Q_8 \succ Q_9 | Q_5 \rangle$ and $\langle Q_8 \succ Q_{10} | Q_5 \rangle$. Question Q_8 would have been a rephrasing of the reference question, were it not for the noun "art" modifying the focus noun phrase "summer camp". Therefore, the information content of the answer to Q_8 is strictly subsumed in the information content associated with the answer to Q_5. Similarly, in Q_9 the focus noun phrase is further specialized through the prepositional phrase "for girls". Therefore, (an answer to) Q_9 is less *useful* to Q_5 than (an answer to) Q_8, i.e. $\langle Q_8 \succ Q_9 | Q_5 \rangle$. Furthermore, the focus "art summer camp" in Q_8 conceptually subsumes the focus "summer camps for singing" in Q_{10}, therefore $\langle Q_8 \succ Q_{10} | Q_5 \rangle$.

We call this dataset *simple* since most of the reference questions are shorter than the other questions in their group. We have also created a *complex* version of the same dataset, by selecting as the reference question in each group a longer question from the same group. For example, if Q_0 were a reference question, it would be replaced with a more complex question, such as Q_2, or Q_3. The annotation is redone to reflect the relative usefulness relations with respect to the new reference questions. We believe that the new *complex* dataset is closer to the actual distribution of questions in community QA repositories: unanswered questions tend to be more specific (longer), whereas general questions (shorter) are more likely to have been answered already. Each dataset is annotated by two annotators, leading to a total of 4 datasets: Simple$_1$, Simple$_2$, Complex$_1$, and Complex$_2$.

Table 2 presents the following statistics on the two types of datasets (Simple, Complex) for each annotator (1, 2): the total number of paraphrasings ($\mathcal{P}$), the total number of useful questions ($\mathcal{U}$), the total number of neutral questions ($\mathcal{N}$), the total number of *more useful than* ordered pairs encoded in the dataset, either explicitly or through transitivity, and the Inter-Annotator Agreement (ITA). We compute the ITA as the *precision* (*P*) and *recall* (*R*) with respect to the *more useful than* ordered pairs encoded in one annotation ($Pairs_1$) relative to the ordered pairs encoded in the other annotation ($Pairs_2$).

Dataset	$\mathcal{P}$	$\mathcal{U}$	$\mathcal{N}$	Pairs	ITA
Simple$_1$	164	775	594	11015	*P*: 76.6
Simple$_2$	134	778	621	10436	*R*: 81.6
Complex$_1$	103	766	664	10654	*P*: 71.3
Complex$_2$	89	730	714	9979	*R*: 81.3

Table 2: Dataset statistics.

$$P = \frac{|Pairs_1 \cap Pairs_2|}{Pairs_1} \quad R = \frac{|Pairs_1 \cap Pairs_2|}{Pairs_2}$$

The statistics in Table 2 indicate that the second annotator was in general more conservative in tagging questions as paraphrases or useful questions.

3 Unsupervised Methods for Question Ranking

An ideal question ranking method would take an arbitrary triplet of questions Q_r, Q_i and Q_j as input, and output an ordering between Q_i and Q_j with respect to the reference question Q_r, i.e. one of $\langle Q_i \succ Q_j | Q_r \rangle$, $\langle Q_i = Q_j | Q_r \rangle$, or $\langle Q_j \succ Q_i | Q_r \rangle$. One approach is to design a *usefulness* function $u(Q_i, Q_r)$ that measures how useful question Q_i is for the reference question Q_r, and define the *more useful than* ($\succ$) relation as follows:

$$\langle Q_i \succ Q_j | Q_r \rangle \Leftrightarrow u(Q_i, Q_r) > u(Q_j, Q_r)$$

If we define $I(Q)$ to be the *information need* associated with question Q, then $u(Q_i, Q_r)$ could be defined as a measure of the relative overlap between $I(Q_i)$ and $I(Q_r)$. Unfortunately, the information need is a concept that, in general, is defined only intensionally and therefore it is difficult to measure. For lack of an operational definition of the information need, we will approximate $u(Q_i, Q_r)$ directly as a measure of the similarity between Q_i and Q_r. The similarity between two questions can be seen as a special case of text-to-text similarity, consequently one possibility is to use a general text-to-text similarity function such as *cosine similarity* in the vector space model (Baeza-Yates and Ribeiro-Neto, 1999):

$$cos(Q_i, Q_r) = \frac{Q_i^T Q_r}{\|Q_i\| \|Q_r\|}$$

Here, Q_i and Q_r denote the corresponding *tf*×*idf* vectors.

As a measure of question similarity, one major drawback of cosine similarity is that it is oblivious of the meanings of words in each question. This particular problem is illustrated by the three questions below. Q_{22} and Q_{23} have the same cosine similarity with Q_{21}, they are therefore indistinguishable in terms of their usefulness to the reference question Q_{21}, even though we expect Q_{22} to be more useful than Q_{23} (a place that sells hydrangea often sells other types of plants too, possibly including cacti).

Q_{21} Where can I buy a hydrangea?

Q_{22} Where can I buy a cactus?

Q_{23} Where can I buy an iPad?

To alleviate the lexical chasm, we can redefine $u(Q_i, Q_r)$ to be the similarity measure proposed by (Mihalcea et al., 2006) as follows:

$$mcs(Q_i, Q_r) = \frac{\sum_{w \in \{Q_i\}} maxSim(w, Q_r) * idf(w)}{\sum_{w \in \{Q_i\}} idf(w)} + \frac{\sum_{w \in \{Q_r\}} maxSim(w, Q_i) * idf(w)}{\sum_{w \in \{Q_r\}} idf(w)}$$

Since scaling factors are immaterial for ranking, we have ignored the normalization constant contained in the original measure. For each word $w \in Q_i$, $maxSim(w, Q_r)$ computes the maximum semantic similarity between w and any word $w_r \in Q_r$. The similarity scores are weighted by the corresponding *idf*'s, and normalized. A similar score is computed for each word $w \in Q_r$. The score computed by $maxSim$ depends on the actual function used to compute the word-to-word semantic similarity. In this paper, we evaluated four of the knowledge-based measures explored in (Mihalcea et al., 2006): *wup* (Wu and Palmer, 1994), *res* (Resnik, 1995), *lin* (Lin, 1998), and *jcn* (Jiang and Conrath, 1997).

4 Supervised Learning for Question Ranking

Cosine similarity, henceforth referred as *cos*, treats questions as bags-of-words. The meta-measure proposed in (Mihalcea et al., 2006), henceforth called *mcs*, treats questions as bags-of-concepts. Both *cos* and *mcs* ignore the syntactic relations between the words in a question, and therefore may miss important structural information. In the next three sections we describe a set of structural features that we believe are relevant for judging question similarity. These and other types of features will be integrated in an SVM model for ranking, as described later in Section 4.4.

4.1 Matching the Focus Words

If we consider the question Q_{24} below as reference, question Q_{26} will be deemed more useful than Q_{25} when using *cos* or *mcs* because of the higher relative lexical and conceptual overlap with Q_{24}. However, this is contrary to the actual ordering $\langle Q_{25} \succ Q_{26} | Q_{24} \rangle$, which reflects the fact that Q_{25}, which expects the same answer type as Q_{24}, should be deemed more useful than Q_{26}, which has a different answer type.

Q_{24} What are some good thriller *movies*?

Q_{25} What are some thriller *movies* with happy ending?

Q_{26} What are some good *songs* from a thriller movie?

The analysis above shows the importance of using the answer type when computing the similarity between two questions. However, instead of relying exclusively on a predefined hierarchy of answer types, we identify the *question focus* of a question, defined as the set of maximal noun phrases in the question that corefer with the expected answer (Bunescu and Huang, 2010a). Focus nouns such as *movies* and *songs* provide more discriminative information than general answer types such as *products*. We use answer types only for questions such as Q_{27} or Q_{28} below that lack an explicit question focus. In such cases, an artificial question focus is created from the answer type (e.g. *location* for Q_{27}, or *method* for Q_{28}).

Q_{27} *Where* can I buy a good coffee maker?

Q_{28} *How* do I make a pizza?

Let f_i and f_r be the focus words corresponding to questions Q_i and Q_r. We introduce a focus feature ϕ_f, and set its value to be equal with the similarity between the focus words:

$$\phi_f(Q_i, Q_r) = wsim(f_i, f_r) \qquad (1)$$

We use *wsim* to denote a generic word meaning similarity measure (e.g. *wup*, *res*, *lin* or *jcn*). When computing the focus feature, the non-focus word "movie" in Q_{26} will not be compared with the focus word "movies" in Q_{24}, and therefore Q_{26} will have a lower value for this feature than Q_{25}, i.e. $\phi_f(Q_{26}, Q_{24}) < \phi_f(Q_{25}, Q_{24})$.

4.2 Matching the Main Verbs

In addition to the question focus, the *main verb* of a question can also provide key information in estimating question-to-question similarity. We define the main verb to be the content verb that is highest in the dependency tree of the question, e.g. *buy* for Q_{27}, or *make* for Q_{28}. If the question does not contain a content verb, the main verb is defined to be the highest verb in the dependency tree, as for example *are* in Q_{24} to Q_{26}. The utility of a question's main verb in judging its similarity to other questions can be seen more clearly in the questions below, where Q_{29} is the reference:

Q_{29} How can I *transfer* music from iTunes to my iPod?

Q_{30} How can I *upload* music to my iPod?

Q_{31} How can I *play* music in iTunes?

The fact that *upload*, as the main verb of Q_{30}, is more semantically related to *transfer* is essential in deciding that $\langle Q_{30} \succ Q_{31} | Q_{29} \rangle$, i.e. Q_{30} is more useful than Q_{31} to Q_{29}.

Let v_i and v_r be the main verbs corresponding to questions Q_i and Q_r. We introduce a main verb feature ϕ_v as follows:

$$\phi_v(Q_i, Q_r) = wsim(v_i, v_r) \qquad (2)$$

If Q_{29} is considered as reference question, it is expected that the main verb feature for question Q_{30} will have a higher value than the main verb feature for Q_{31}, i.e. $\phi_f(Q_{31}, Q_{29}) < \phi_f(Q_{30}, Q_{29})$.

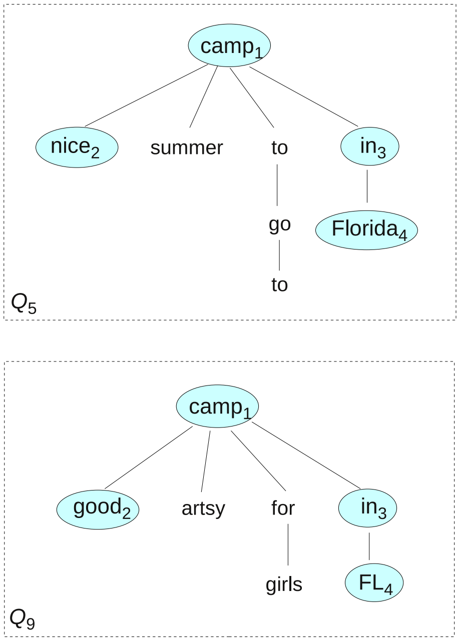


Figure 2: Matched dependency trees.

4.3 Matching the Dependency Trees

The question focus and the main verb are only two of the nodes in the syntactic dependency tree of a question. In general, all the words in a question are important when judging its semantic similarity with another question. We therefore propose a more general feature that exploits the dependency structure of the question and, in doing so, it also considers all the words in the question, like *cos* and *mcs*. For any given question we initially ignore the direction of the dependency arcs and change the question dependency tree to be rooted at the focus word, as illustrated in Figure 2 for questions Q_5 and Q_9. Interrogative patterns such as "What is" or "Are there any" are automatically eliminated from the dependency trees. We define the dependency tree similarity between two questions Q_i and Q_r to be a function of similarities $wsim(v_i, v_r)$ computed between aligned nodes $v_i \in Q_i$ and $v_r \in Q_r$. The nodes of two dependency trees are aligned through a function $\texttt{MaxMatch}(u_i.\mathcal{C},\ u_r.\mathcal{C})$ that takes two sets of children nodes as arguments, one from Q_i and one from Q_r, and finds the maximum weighted bipartite matching between $u_i.\mathcal{C}$ and $u_r.\mathcal{C}$. Given two children nodes $v_i \in u_i.\mathcal{C}$ and $v_r \in u_r.\mathcal{C}$, the weight of a potential matching between v_i and v_r is defined simply as $wsim(v_i, v_r)$. MaxMatch($u_i.\mathcal{C}$, $u_r.\mathcal{C}$) is furthermore constrained to match only nodes that have compatible part-of-speech tags (e.g. nouns are matched to nouns, verbs are matched to verbs), and children nodes that have the same head-modifier relationship with their parents (i.e. they are both heads, or they are both dependents of their parents). Table 3 shows the recursive algorithm used for finding a matching between two question dependency trees rooted at the focus words. The initial arguments of the algorithm are the two focus words $u_i = f_i$ and $u_r = f_r$. Thus, the pair (f_i, f_r) is the first pair of nodes to be added to the matching $\mathcal{M}$ in step 1. In the next step, we compute the maximum weighted matching between the children nodes $u_i.\mathcal{C}$ and $u_r.\mathcal{C}$, and recursively call the matching algorithm on pairs of matched nodes (v_i, v_r) from $\mathcal{M}$. The algorithm stops when MaxMatch returns an empty matching, which may happen when reaching leaf nodes, or when no pair of children nodes has compatible POS tags, or child-parent dependencies. Figure 2 shows the results of applying the tree matching algorithm on questions Q_5 and Q_9. Matched nodes share the same index and are shown in circles, whereas unmatched nodes are shown in italics.

```
TreeMatch($u_i, u_r$)
[In]: Two dependency tree nodes $u_i$, $u_r$.
[Out]: A set of node pairs $\mathcal{M}$.
1. set $\mathcal{M} \leftarrow \{(u_i, u_r)\}$
2. for each $(v_i, v_r) \in$ MaxMatch($u_i.\mathcal{C}, u_r.\mathcal{C}$):
3.     set $\mathcal{M} \leftarrow \mathcal{M} \cup$ TreeMatch($v_i, v_r$)
4. return $\mathcal{M}$
```

Table 3: Dependency Tree Matching.

We introduce a new feature $\phi_t(Q_i, Q_r)$ whose value is defined as the dependency tree similarity between questions Q_i and Q_r. Once the optimum matching $\mathcal{M}(Q_i, Q_r)$ between dependency trees has been found, $\phi_t(Q_i, Q_r)$ is computed as the normalized sum of the similarities between pairs of matched nodes v_i and v_r, as shown in Equations 3 and 4 below. When computing the similarity between two matched nodes, we factor in the similarities between corresponding pairs of words on the paths $f_i \rightsquigarrow v_i$, $f_r \rightsquigarrow v_r$ between the focus words f_i, f_r and the nodes v_i, v_r, as shown in Equation 5. This has the effect of reducing the importance of words that are farther away from the focus word in the dependency tree.

$$\phi_t(Q_i, Q_r) = \frac{sim(Q_i, Q_r)}{\sqrt{sim(Q_i, Q_i) sim(Q_r, Q_r)}} \tag{3}$$

$$sim(Q_i, Q_r) = \sum_{(v_i, v_r) \in \mathcal{M}(Q_i, Q_r)} sim(f_i \rightsquigarrow v_i, f_r \rightsquigarrow v_r) \tag{4}$$

$$sim(u_1 \rightsquigarrow u_n, v_1 \rightsquigarrow v_n) = \prod_{i=1}^{n} wsim(u_i, v_i) \tag{5}$$

If the word similarity function is normalized and defined to return 1 for identical words, the normalizer in Equation 3 becomes equivalent with $\sqrt{|Q_i||Q_r|}$. Thus, words that are left unmatched implicitly decrease the dependency tree similarity.

4.4 An SVM Model for Ranking Questions

We consider learning a *usefulness* function $u(Q_i, Q_r)$ of the following general, linear form:

$$u(Q_i, Q_r) = \mathbf{w}^T \phi(Q_i, Q_r) \tag{6}$$

The vector $\phi(Q_i, Q_r)$ is defined to contain the following generic features:

1. $\phi_f(Q_i, Q_r)$ = the semantic similarity between focus words, as described in Section 4.1.
2. $\phi_v(Q_i, Q_r)$ = the semantic similarity between main verbs, as described in Section 4.2.
3. $\phi_t(Q_i, Q_r)$ = the semantic similarity between the dependency trees, as described in Section 4.3.
4. $cos(Q_i, Q_r)$ = the cosine similarity between the two questions, as described in Section 3.
5. $mcs(Q_i, Q_r)$ = the bag-of-concepts similarity between the two questions, as described in Section 3.

Each of the generic features ϕ_f, ϕ_v, ϕ_t, and mcs corresponds to four actual features, one for each possible choice of the word similarity function $wsim$ (i.e. *wup*, *res*, *lin* or *jcn*). An additional pair of features is targeted at questions containing locations:

6. $\phi_l(Q_i, Q_r)$ = 1 if both questions contain locations, 0 otherwise.

7. $\phi_d(Q_i, Q_r)$ = the normalized geographical distance between the locations in Q_i and Q_r, 0 if $\phi_l(Q_i, Q_r)$ = 0.

Given two location names, we first find their latitude and longitude using Google Maps, and then compute the spherical distance between them using the *haversine formula*.

The corresponding parameters $\mathbf{w}$ will be trained on pairs from one of the partially ordered datasets described in Section 2. We use the kernel version of the large-margin ranking approach from (Joachims, 2002) which solves the optimization problem in Figure 3 below. The aim of this formulation is to find a weight vector $\mathbf{w}$ such that 1) the number of ranking constraints $u(Q_i, Q_r) \geq u(Q_j, Q_r)$ from the training data $\mathcal{D}$ that are violated is minimized, and 2) the ranking function $u(Q_i, Q_r)$ generalizes well beyond the training data. The learned $\mathbf{w}$ is a linear combination of the feature vectors $\phi(Q_i, Q_r)$, which makes it possible to use kernels.

minimize:
$$J(w, \xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum \xi_{rij}$$
subject to:
$$\mathbf{w}^T\phi(Q_i, Q_r) - \mathbf{w}^T\phi(Q_j, Q_r) \geq 1 - \xi_{rij}$$
$$\xi_{rij} \geq 0$$
$$\forall Q_r, Q_i, Q_j \in \mathcal{D}, \langle Q_i \succ Q_j | Q_r\rangle$$

Figure 3: SVM ranking optimization problem.

5 Experimental Evaluation

We use the four question ranking datasets described in Section 2 to evaluate the three similarity measures cos, mcs, and ϕ_t, as well as the SVM ranking model. We report one set of results for each of the four word similarity measures *wup*, *res*, *lin* or *jcn*. Each question similarity measure is evaluated in terms of its accuracy on the set of ordered pairs, and the performance is averaged between the two annotators for the *Simple* and *Complex* datasets. If $\langle Q_i \succ Q_j | Q_r\rangle$ is a relation specified in the annotation, we consider the tuple $\langle Q_i, Q_j, Q_r\rangle$ correctly classified if and only if $u(Q_i, Q_r) > u(Q_j, Q_r)$, where u is the question similarity measure. We used the SVM^{light} [2] implementation of ranking SVMs, with a cubic kernel and the standard parameters. The SVM ranking model was trained and tested using 10-fold cross-validation, and the overall accuracy was computed by averaging over the 10 folds.

We used the NLTK [3] implementation of the four similarity measures *wup*, *res*, *lin* or *jcn*. The *idf* values for each word were computed from frequency counts over the entire Wikipedia. For each question, the *focus* is identified automatically by an SVM tagger trained on a separate corpus of 2,000 questions manually annotated with focus information (Bunescu and Huang, 2010a). The SVM tagger uses a combination of lexico-syntactic features and a quadratic kernel to achieve a 93.5% accuracy in a 10-fold cross validation evaluation on the 2,000 questions. The head-modifier dependencies were derived automatically from the syntactic parse tree using the head finding rules from (Collins, 1999). The syntactic tree is obtained using Spear [4], a syntactic parser which comes pre-trained on an additional treebank of questions. The *main verb* of a question is identified deterministically using a breadth first traversal of the dependency tree.

The overall accuracy results presented in Table 4 show that the SVM ranking model obtains by far the best performance on both datasets, a substantial 10% higher than *cos*, which is the best performing unsupervised method. The random baseline – assigning a random similarity value to each pair of questions – results in 50% accuracy. Even though its use of word senses was expected to lead to superior results, *mcs* does not perform better than *cos* on this dataset. Our implementation of *mcs* did however perform better than *cos* on the Microsoft paraphrase corpus (Dolan et al., 2004). One possible reason for this behavior is that *mcs* seems to be less resilient than *cos* to differences in question length. Whereas the Microsoft paraphrase corpus was specifically designed such that "the length of the shorter of the two sentences, in words, is at least 66% that of the longer" (Dolan and Brockett, 2005), the question ranking datasets place no constraints on the lengths of the

[2] svmlight.joachims.org

[3] www.nltk.org

[4] www.surdeanu.name/mihai/spear

Question		*wup*		*res*		*lin*		*jcn*		
Dataset	cos	mcs	ϕ_t	mcs	ϕ_t	mcs	ϕ_t	mcs	ϕ_t	*SVM*
Simple	73.7	69.1	69.4	71.3	71.8	70.8	69.8	71.9	71.7	**82.1**
Complex	72.6	64.1	69.6	66.0	71.5	66.9	69.1	69.4	71.0	**82.5**

Table 4: Pairwise accuracy results.

Dataset	all	$-\phi_f$	$-\phi_v$	$-\phi_t$	$-\phi_{l,d}$	$-cos$	$-mcs$	$-\phi_{f,t}$
Simple	**82.1**	79.3	82.0	80.2	81.5	80.3	81.4	78.5
Complex	**82.5**	81.3	81.3	78.7	81.8	79.2	81.8	77.4

Table 5: Ablation results.

questions. However, even though by themselves the meaning aware mcs and the structure-and-meaning aware ϕ_t do not outperform the bag-of-words cos, they do help in increasing the performance of the SVM ranking model, as can be inferred from the corresponding columns in Table 5. The table shows the results of ablation experiments in which all but one type of features are used. The results indicate that all types of features are useful, with significant contributions being brought especially by cos and the focus related features $\phi_{f,t}$.

The measures investigated in this paper are all compositional and reduce the similarity computations to word level. The following question patterns illustrate the need to design more complex similarity measures that take into account the context of every word in the question:

P_1 Where can I find a job around ⟨*City* ⟩?

P_2 What are some famous people from ⟨*City* ⟩?

P_3 What is the population of ⟨*City* ⟩?

Below are three instantiations of the first question pattern:

Q_{32} Where can I find a job around Anaheim, CA?

Q_{33} Where can I find a job around Los Angeles?

Q_{34} Where can I find a job around Vista, CA?

If we take Q_{32} as reference question, the fact that the distance between Los Angeles and Anaheim is smaller than the distance between Vista and Anaheim leads the ranking system to rank Q_{33} as more useful than Q_{34} with respect to Q_{32}, which is the expected result. The preposition "around" from the city context in the first pattern is a good indicator that proximity relations are relevant in this case. When the same three cities are used for instantiating the other two patterns, it can be seen that the proximity relations are no longer as relevant for judging the relative usefulness of questions.

6 Future Work

We plan to integrate context dependent word similarity measures into a more robust question utility function. We also plan to make the dependency tree matching more flexible in order to account for paraphrase patterns that may differ in their syntactic structure. The questions that are posted on community QA sites often contain spelling or grammatical errors. Consequently, we will work on interfacing the question ranking system with a separate module aimed at fixing orthographic and grammatical errors.

7 Related Work

The question rephrasing subtask has spawned a diverse set of approaches. (Hermjakob et al., 2002) derive a set of phrasal patterns for question reformulation by generalizing surface patterns acquired automatically from a large corpus of web documents. The focus of the work in (Tomuro, 2003) is on deriving reformulation patterns for the interrogative part of a question. In (Jeon et al., 2005), word translation probabilities are trained on pairs of semantically similar questions that are automatically extracted from an FAQ archive, and then used in a language model that retrieves question reformulations. (Jijkoun and de Rijke, 2005) describe an FAQ question retrieval system in which weighted combinations of similarity functions corresponding to questions, existing answers, FAQ titles and pages are computed using a vector space model. (Zhao et al., 2007) exploit the Encarta logs to automatically extract clusters containing question paraphrases and further train a perceptron to recognize question paraphrases inside each cluster based on a combination of lexical, syntactic and semantic similarity features. More recently, (Bernhard and Gurevych, 2008) evaluated various string similarity measures and vector space based similarity measures on the task of retrieving question paraphrases from the WikiAnswers repository. The aim of the question search task presented in (Duan et al., 2008) is to return questions that are semantically equivalent or close to the queried question, and is therefore similar to our question ranking task. Their approach is evaluated on a dataset in which questions are categorized either as relevant or irrelevant. Our formulation of question ranking is more general, and in particular subsumes the annotation of binary question categories such as relevant vs. irrelevant, or paraphrases vs. non-paraphrases. Moreover, we are able to exploit the annotated utility relations as supervision in a learning for ranking approach, whereas (Duan et al., 2008) use the annotated dataset to tune the 3 parameters of a mostly unsupervised approach. The question ranking task was first formulated in (Bunescu and Huang, 2010b), where an initial version of the dataset was also described. In this paper, we introduce 4 versions of the dataset, a more general meaning and structure aware similarity measure, and a supervised model for ranking that substantially outperforms the previously proposed utility measures.

8 Conclusion

We presented a supervised learning approach to the question ranking task in which previously known questions are ordered based on their relative utility with respect to a new, reference question. We created four versions of a dataset of 60 groups of questions [5], each annotated with a partial order relation reflecting the relative utility of questions inside each group. An SVM ranking model was trained on the dataset and evaluated together with a set of simpler, unsupervised question-to-question similarity models. Experimental results demonstrate the importance of using structure and meaning aware features when computing the relative usefulness of questions.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments.

References

Ricardo Baeza-Yates and Berthier Ribeiro-Neto. 1999. *Modern Information Retrieval*. ACM Press, New York.

Delphine Bernhard and Iryna Gurevych. 2008. Answering learners' questions by retrieving question paraphrases from social Q&A sites. In *EANL '08: Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications*, pages 44–52, Morristown, NJ, USA. Association for Computational Linguistics.

Razvan Bunescu and Yunfeng Huang. 2010a. Towards a general model of answer typing: Question focus identification. In *Proceedings of The 11th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2010), RCS Volume*, pages 231–242.

Razvan Bunescu and Yunfeng Huang. 2010b. A utility-driven approach to question ranking in social QA. In *Proceedings of The 23rd International Conference on Computational Linguistics (COLING 2010)*, pages 125–133.

Michael Collins. 1999. *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, pages 9–16.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting assively parallel news sources. In *Proceedings of The 20th International Conference on Computational Linguistics (COLING'04)*, page 350.

Huizhong Duan, Yunbo Cao, Chin-Yew Lin, and Yong Yu. 2008. Searching questions by identifying question topic and question focus. In *Proceedings of ACL-08: HLT*, pages 156–164, Columbus, Ohio, June.

Ulf Hermjakob, Abdessamad Echihabi, and Daniel Marcu. 2002. Natural language based reformulation

[5] The dataset will be made publicly available.

resource and web exploitation for question answering. In *Proceedings of TREC-2002*.

Jiwoon Jeon, W. Bruce Croft, and Joon Ho Lee. 2005. Finding similar questions in large question and answer archives. In *Proceedings of the 14th ACM international conference on Information and knowledge management (CIKM'05)*, pages 84–90, New York, NY, USA. ACM.

J.J. Jiang and D.W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference on Research in Computational Linguistics*, pages 19–33.

Valentin Jijkoun and Maarten de Rijke. 2005. Retrieving answers from frequently asked questions pages on the Web. In *Proceedings of the 14th ACM international conference on Information and knowledge management (CIKM'05)*, pages 76–83, New York, NY, USA. ACM.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)*, Edmonton, Canada.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML '98)*, pages 296–304, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the 21st national conference on Artificial intelligence (AAAI'06)*, pages 775–780. AAAI Press.

Dan I. Moldovan, Marius Pasca, Sanda M. Harabagiu, and Mihai Surdeanu. 2002. Performance issues and error analysis in an open-domain question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 33–40, Philadelphia, PA, July.

John M. Prager. 2006. Open-domain question-answering. *Foundations and Trends in Information Retrieval*, 1(2):91–231.

Philip Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI'95: Proceedings of the 14th international joint conference on Artificial intelligence*, pages 448–453, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Noriko Tomuro. 2003. Interrogative reformulation patterns and acquisition of question paraphrases. In *Proceedings of the Second International Workshop on Paraphrasing*, pages 33–40, Morristown, NJ, USA. Association for Computational Linguistics.

Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138, Morristown, NJ, USA. Association for Computational Linguistics.

Shiqi Zhao, Ming Zhou, and Ting Liu. 2007. Learning question paraphrases for QA from Encarta logs. In *Proceedings of the 20th international joint conference on Artifical intelligence (IJCAI'07)*, pages 1795–1800, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Positional Language Models for Clinical Information Retrieval

Florian Boudin
DIRO, Université de Montréal
CP. 6128, succ. Centre-ville
H3C 3J7 Montréal, Canada
boudinfl@iro.umontreal.ca

Jian-Yun Nie
DIRO, Université de Montréal
CP. 6128, succ. Centre-ville
H3C 3J7 Montréal, Canada
nie@iro.umontreal.ca

Martin Dawes
Department of Family Medicine
McGill University, 515 Pine Ave
H2W 1S4 Montréal, Canada
martin.dawes@mcgill.ca

Abstract

The PECO framework is a knowledge representation for formulating clinical questions. Queries are decomposed into four aspects, which are Patient-Problem (P), Exposure (E), Comparison (C) and Outcome (O). However, no test collection is available to evaluate such framework in information retrieval. In this work, we first present the construction of a large test collection extracted from systematic literature reviews. We then describe an analysis of the distribution of PECO elements throughout the relevant documents and propose a language modeling approach that uses these distributions as a weighting strategy. In our experiments carried out on a collection of 1.5 million documents and 423 queries, our method was found to lead to an improvement of 28% in MAP and 50% in P@5, as compared to the state-of-the-art method.

1 Introduction

In recent years, the volume of health and biomedical literature available in electronic form has grown exponentially. MEDLINE, the authoritative repository of citations from the medical and bio-medical domain, contains more than 18 million citations. Searching for clinically relevant information within this large amount of data is a difficult task that medical professionals are often unable to complete in a timely manner. A better access to clinical evidence represents a high impact application for physicians.

Evidence-Based Medicine (EBM) is a widely accepted paradigm for medical practice (Sackett et al., 1996). EBM is defined as the conscientious, explicit and judicious use of current best evidence in making decisions about patient care. Practice EBM means integrating individual clinical expertise with the best available external clinical evidence from systematic research. It involves tracking down the best evidence from randomized trials or meta-analyses with which to answer clinical questions. Richardson et al. (1995) identified the following four aspects as the key elements of a well-built clinical question:

- **Patient-problem**: what are the patient characteristics (e.g. age range, gender, etc.)? What is the primary condition or disease?
- **Exposure-intervention**: what is the main intervention (e.g. drug, treatment, duration, etc.)?
- **Comparison**: what is the exposure compared to (e.g. placebo, another drug, etc.)?
- **Outcome**: what are the clinical outcomes (e.g. healing, morbidity, side effects, etc.)?

These elements are known as the PECO elements. Physicians are educated to formulate their clinical questions in respect to this structure. For example, in the following question: "*In patients of all ages with Parkinson's disease, does a Treadmill training compared to no training allows to increase the walking distance*?" one can identify the following elements:

- **P**: Patients of all ages with Parkinson's disease
- **E**: Treadmill training
- **C**: No treadmill training
- **O**: Walking distance

In spite of this well-defined question structure, physicians still use keyword-based queries when they search for clinical evidence. An explanation of

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 108–115,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

that is the almost total absence of PECO search interfaces. PubMed[1], the most used search interface, does not allow users to formulate PECO queries yet. For the previously mentioned clinical question, a physician would use the query "*Treadmill* AND *Parkinson's disease*". There is intuitively much to gain by using a PECO structured query in the retrieval process. This structure specifies the role of each concept in the desired documents, which is a clear advantage over a keyword-based approach. One can for example differentiate two queries in which a disease would be a patient condition or a clinical outcome. This conceptual decomposition of queries is also particularly useful in a sense that it can be used to balance the importance of each element in the search process.

Another important factor that prevented researchers from testing approaches to clinical information retrieval (IR) based on PECO elements is the lack of a test collection, which contains a set of documents, a set of queries and the relevance judgments. The construction of such a test collection is costly in manpower. In this paper, we take advantage of the systematic reviews about clinical questions from Cochrane. Each Cochrane review examines in depth a clinical question and survey all the available relevant publications. The reviews are written for medical professionals. We transformed them into a TREC-like test collection, which contains 423 queries and 8926 relevant documents extracted from MEDLINE. In a second part of this paper, we present a model integrating the PECO framework in a language modeling approach to IR. An intuitive method would try to annotate the concepts in documents into PECO categories. One can then match the PECO elements in the query to the elements detected in documents. However, as previous studies have shown, it is very difficult to automatically annotate accurately PECO elements in documents. To by-pass this issue, we propose an alternative that relies on the observed positional distribution of these elements in documents. We will see that different types of element have different distributions. By weighting words according to their positions, we can indirectly weigh the importance of different types of element in search. As we will show in this paper, this approach turns out to be highly effective.

This paper is organized as follows. We first briefly review the previous work, followed by a description of the test collection we have constructed. Next, we give the details of the method we propose and present our experiments and results. Lastly, we conclude with a discussion and directions for further work.

2 Related work

The need to answer clinical questions related to a patient care using IR systems has been well studied and documented (Hersh et al., 2000; Niu et al., 2003; Pluye et al., 2005). There are a limited but growing number of studies trying to use the PECO elements in the retrieval process. (Demner-Fushman and Lin, 2007) is one of the few such studies, in which a series of knowledge extractors is used to detect PECO elements in documents. These elements are later used to re-rank a list of retrieved citations from PubMed. Results reported indicate that their method can bring relevant citations into higher-ranking positions, and from these abstracts generate responses that answer clinicians' questions. This study demonstrates the value of the PECO framework as a method for structuring clinical questions. However, as the focus has been put on the post-retrieval step (for question-answering), it is not clear whether PECO elements are useful at the retrieval step. Intuitively, the integration of PECO elements in the retrieval process can also lead to higher retrieval effectiveness.

The most obvious scenario for testing this would be to recognize PECO elements in documents prior to indexing. When a PECO-structured query is formulated, it is matched against the PECO elements in the documents (Dawes et al., 2007). Nevertheless, the task of automatically identifying PECO elements is a very difficult one. There are two major reasons for that. First, previous studies have indicated that there is a low to moderate agreement rate among humans for annotating PECO elements. This is due to the lack of standard definition for the element' boundaries (e.g. can be words, phrases or sentences) but also to the existence of several levels of annotation. Indeed, there are a high number

[1] `www.pubmed.gov`

of possible candidates for each element and one has to choose if it is a main element (i.e. playing a major role in the clinical study) or secondary elements. Second is the lack of sufficient annotated data that can be used to train automatic tagging tools.

Despite all these difficulties, several efficient detection methods have been proposed (Demner-Fushman and Lin, 2007; Chung, 2009). Nearly all of them are however restricted to a coarse-grain annotation level (i.e. tagging entire sentences as describing one element). This kind of coarser-grain identification is more robust and more feasible than the one at concept level, and it could be sufficient in the context of IR. In fact, for IR purposes, what is the most important is to correctly weight the words in documents and queries. From this perspective, an annotation at the sentence level may be sufficient. Notwithstanding, experiments conducted using a collection of documents that were annotated at a sentence-level only showed a small increase in retrieval accuracy (Boudin et al., 2010b) compared to a traditional bag-of-words approach.

More recently, Boudin et al. (2010a) proposed an alternative to the PECO detection issue that relies on assigning different weights to words according to their positions in the document. A location-based weighting strategy is used to emphasize the most informative parts of documents. They show that a large improvement in retrieval effectiveness can be obtained this way and indicate that the weights learned automatically are correlated to the observed distribution of PECO elements in documents. In this work, we propose to go one step further in this direction by analyzing the distribution of PECO elements in a large number of documents and define the positional probabilities of PECO elements accordingly. These probabilities will be integrated in the document language model.

3 Construction of the test collection

Despite the increasing use of search engines by medical professionals, there is no standard test collection for evaluating clinical IR. Constructing such a resource from scratch would require considerable time and money. One way to overcome this obstacle is to use already available systematic reviews. Systematic reviews try to identify, appraise, select and synthesize all high quality research evidence relevant to a clinical question. The best-known source of systematic reviews in the healthcare domain is the Cochrane collaboration[2]. It consists of a group of over 15,000 specialists who systematically identify and review randomized trials of the effects of treatments. In particular, a review contains a reference section, listing all the relevant studies to the clinical question. These references can be considered as relevant documents. In our work, we propose to use these reviews as a way to semi-automatically build a test collection. As the reviews are made by specialists in the area independently from our study, we can avoid bias in our test collection.

We gathered a subset of Cochrane systematic reviews and asked a group of annotators, one professor and four Master students in family medicine, to create PECO-structured queries corresponding to the clinical questions. As clinical questions answered in these reviews cover various aspects of one topic, multiple variants of precise PECO queries were generated for each review. Moreover, in order to be able to compare a PECO-based search strategy to a real world scenario, this group have also provided the keyword-based queries that they would have used to search with PubMed. Below is an example of queries generated from the systematic review about "*Aspirin with or without an antiemetic for acute migraine headaches in adults*":

Keyword-based query

[aspirin and migraine]

PECO-structured queries

1. [adults 18 years or more with migraine]$^{\mathrm{P}}$
   [aspirin alone]$^{\mathrm{E}}$
   [placebo]$^{\mathrm{C}}$
   [pain free]$^{\mathrm{O}}$
2. [adults 18 years or more with migraine]$^{\mathrm{P}}$
   [aspirin plus an antiemetic]$^{\mathrm{E}}$
   [placebo]$^{\mathrm{C}}$
   [pain free]$^{\mathrm{O}}$
3. [adults 18 years or more with migraine]$^{\mathrm{P}}$
   [aspirin plus metoclopramide]$^{\mathrm{E}}$
   [active comparator]$^{\mathrm{C}}$
   [use of rescue medication]$^{\mathrm{O}}$

[2] www.cochrane.org

All the citations included in the "References" section of the systematic review were extracted and selected as relevant documents. These citations were manually mapped to PubMed unique identifiers (PMID). This is a long process that was undertaken by two different workers to minimize the number of errors. At this step, only articles published in journals referenced in PubMed are considered (e.g. conference proceedings are not included).

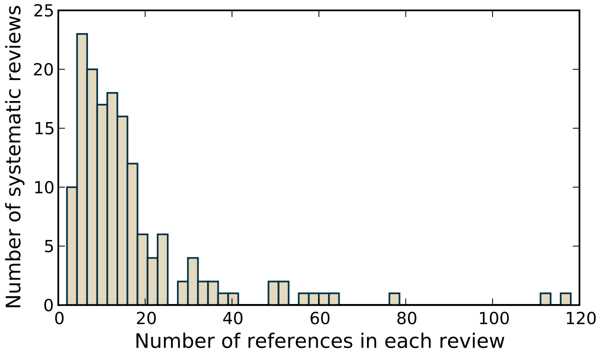


Figure 1: Histogram of the number of queries versus the number of relevant documents.

We selected in sequential order from the set of new systematic reviews[3] and processed 156 Cochrane reviews. There was no restriction about the topics covered or the number of included references. The resulting test collection is composed of 423 queries and 8926 relevant citations (2596 different citations). This number reduces to 8138 citations once we remove the citations without any text in the abstract (i.e. certain citations, especially old ones, only contain a title). Figure 1 shows the statistics derived from the number of relevant documents by query. In this test collection, the average number of documents per query is approximately 19 while the average length of a document is 246 words.

4 Distribution of PECO elements

The observation that PECO elements are not evenly distributed throughout the documents is not new. In fact, most existing tagging methods used location-based features. This information turns out to be very useful because of the standard structure of medical citations. Actually, many scientific journals explicitly recommend authors to write their abstracts in compliance to the ordered rhetorical structure: Introduction, Methods, Results and Discussion. These rhetorical categories are highly correlated to the distributions of PECO elements, as some elements are more likely to occur in certain categories (e.g. clinical outcomes are more likely to appear in the conclusion). The position is thus a strong indicator of whether a text segment contains a PECO element or not.

To the best of our knowledge, the first analysis of the distribution of PECO elements in documents was described in(Boudin et al., 2010a). A small collection of manually annotated abstracts was used to compute the probability that a PECO element occurs in a specific part of the documents. This study is however limited by the small number of annotated documents (approximately 50 citations) and the moderate agreement rate among human annotators. Here we propose to use our test collection to compute more reliable statistics.

The idea is to use the pairs of PECO-structured query and relevant document, assuming that if a document is relevant then it should contain the same elements as the query. Of course, this is obviously not always the case. Errors can be introduced by synonyms or homonyms and relevant documents may not contain all of the elements described in the query. But, with more than 8100 documents, it is quite safe to say that this method produce fairly reliable results. Moreover, a filtering process is applied to queries removing all non-informative words (e.g. stopwords, numbers, etc.) from being counted.

There are several ways to look at the distribution of PECO elements in documents. One can use the rhetorical structure of abstracts to do that. However, the high granularity level of such analysis would make it less precise for IR purposes. Furthermore, most of the citations available in PubMed are devoid of explicitly marked sections. It is possible to automatically detect these sections but only with a non-negligible error rate (McKnight and Srinivasan, 2003). In our study, we chose to use a fixed number of partitions by dividing documents into parts of equal length. This choice is motivated by its repeatability and ease to implement, but also for comparison with previous studies.

We divided each relevant document into 10 parts of equal length on a word level (from P1 to P10). We

[3] http://mrw.interscience.wiley.com/cochrane/cochrane_clsysrev_new_fs.html

computed statistics on the number of query words that occur in each of these parts. For each PECO element, the distribution of query words among the parts of the documents is not uniform (Figure 2). We observe distinctive distributions, especially for Patient-Problem and Exposure elements, indicating that first and last parts of the documents have higher chance to contain these elements. This gives us a clear and robust indication on which specific parts should be enhanced when searching for a given element. Our proposed model will exploit the typical distributions of PECO elements in documents.

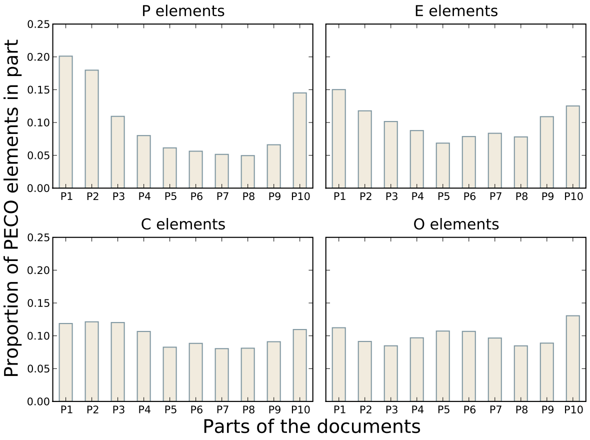


Figure 2: Distribution of each PECO element throughout the different parts of the documents.

5 Retrieval Method

In this work, we use the language modeling approach to information retrieval. This approach assumes that queries and documents are generated from some probability distribution of text (Ponte and Croft, 1998). Under this assumption, ranking a document D as relevant to a query Q is seen as estimating $P(Q|D)$, the probability that Q was generated by the same distribution as D. A typical way to score a document D as relevant to a query Q is to compute the Kullback-Leibler divergence between their respective language models:

$$\mathrm{score}(Q, D) = \sum_{w \in Q} P(w|Q) \cdot \log P(w|D) \qquad (1)$$

Under the traditional bag-of-words assumption, i.e. assuming that there is no need to model term dependence, a simple estimate for $P(w|Q)$ can be obtained by computing Maximum Likelihood Estimation (MLE). It is calculated as the number of times the word w appears in the query Q, divided by its length:

$$P(w|Q) = \frac{\mathrm{count}(w, Q)}{|Q|}$$

A similar method is employed for estimating $P(w|D)$. Bayesian smoothing using Dirichlet priors is however applied to the maximum likelihood estimator to compensate for data sparseness (i.e. smoothing probabilities to remove zero estimates). Given μ the prior parameter and $\mathcal{C}$ the collection of documents, $P(w|D)$ is computed as:

$$P(w|D) = \frac{\mathrm{count}(w, D) + \mu \cdot P(w|\mathcal{C})}{|D| + \mu}$$

5.1 Model definition

In our model, we propose to use the distribution of PECO elements observed in documents to emphasize the most informative parts of the documents. The idea is to get rid of the problem of precisely detecting PECO elements by using a positional language model. To integrate position, we estimate a series of probabilities that constraints the word counts to a specific part of the documents instead of the entire document. Each document D is ranked by a weighted linear interpolation. Given a document D divided in 10 parts $p \in [P1, P2 \cdots P10]$, $P(w|D)$ in equation 1 is redefined as:

$$\begin{aligned} P'(w|D) = \alpha \cdot P(w|D) + \beta \cdot P_{\mathrm{title}}(w|D) \\ + \gamma \cdot \sum_{p_i \in D} \sigma_e \cdot P_{p_i}(w|D) \end{aligned} \qquad (2)$$

where the σ_e weights for each type of element e are empirically fixed to the values of the distribution of PECO elements observed in documents. We then redefine the scoring function to integrate the PECO query formulation. The idea is to use the PECO structure as a way to balance the importance of each element in the retrieval step. The final scoring function is defined as:

$$\mathrm{score}_{\mathrm{final}}(Q, D) = \sum_{e \in \mathrm{PECO}} \delta_e \cdot \mathrm{score}(Q_e, D)$$

In our model, there are a total of 7 weighting parameters, 4 corresponding to the PECO elements in queries (δ_P, δ_E, δ_C and δ_O) and 3 for the document language models (α, β and γ). These parameters will be determined by cross-validation.

6 Results

In this section, we first describe the details of our experimental protocol. Then, we present the results obtained by our model on the constructed test collection.

6.1 Experimental settings

As a collection of documents, we gathered 1.5 millions of citations from PubMed. We used the following constraints: citations with an abstract, human subjects, and belonging to one of the following publication types: randomized control trials, reviews, clinical trials, letters, editorials and meta-analyses. The set of queries and relevance judgments described in Section 3 is used to evaluate our model. Relevant documents were, if not already included, added to the collection. Because each query is generated from a systematic literature review completed at a time t, we placed an additional restriction on the publication date of the retrieved documents: only documents published before time t are considered. Before indexing, each citation is pre-processed to extract its title and abstract text and then converted into a TREC-like document format. Abstracts are divided into 10 parts of equal length (the ones containing less than 10 words are discarded). The following fields are marked in each document: title, P1, P2 $\cdots$ P10. The following evaluation measures are used:

- Precision at rank n (P@n): precision computed on the n topmost retrieved documents.
- Mean Average Precision (MAP): average of precision measures computed at the point of each relevant document in the ranked list.
- Number of relevant documents retrieved

All retrieval tasks are performed using an "out-of-the-shelf" version of the Lemur toolkit[4]. We use the embedded tokenization algorithm along with the standard Porter stemmer. The number of retrieved documents is set to 1000 and the Dirichlet prior smoothing parameter to $\mu = 2000$. In all our experiments, we use the KL divergence scoring function (equation 1) as baseline. Statistical significance is computed using the well-known Student's t-test. To determine reasonable weights and avoid overtuning the parameters, we use a 10-fold cross-validation optimizing the MAP values.

6.2 Experiments

We first investigated the impact of using PECO-structured queries on the retrieval performance. As far as we know, no quantitative evaluation of the increase or decrease of performance in comparison with a keyword-based search strategy has been reported. Schardt et al. (2007) presented a comparison between PubMed and a PECO search interface but failed to demonstrate any significant difference between the two search protocols. The larger number of words in PECO-structured queries, on average 18.8 words per query compared to 4.3 words for keyword queries, should capture more aspects of the information need. But, it may also be a disadvantage due to the fact that more noise can be brought in, causing query-drift issues.

We propose two baselines using the keyword-based queries. The first baseline (named Baseline-1) uses keyword queries with the traditional language modeling approach. This is one of the state-of-the-art approaches in current IR research. This retrieval model considers each word in a query as an equal, independent source of information. In the second baseline (named Baseline-2), we consider multiword phrases. In our test collection, queries are often composed of multiword phrases such as "*low back pain*" or "*early pregnancy*". It is clear that finding the exact phrase "*heart failure*" is a much stronger indicator of relevance than just finding "*heart*" and "*failure*" scattered within a document. The Indri operator `#1` is used to perform phrase-based retrieval. Phrases are already indicated in queries by the conjunction and (e.g. *vaccine and hepatitis B*). A simple regular expression is used to recognize the phrases.

Results are presented in Table 1. As expected, phrase-based retrieval leads to some increase in retrieval precision (P@5). However, the number of

[4] www.lemurproject.org

relevant documents retrieved is decreased. This is due to the fact that we use exact phrase matching that can reduce query coverage. One solution would be to use unordered window features (Indri operator `#uwn`) that would require words to be close together but not necessarily in an exact sequence order (Metzler and Croft, 2005).

The PECO queries use PECO-structured queries as a bag of words. We observe that PECO queries do not enhance the average precision but increase the P@5 significantly. The number of relevant documents retrieved is also larger. These results indicate that formulating clinical queries according to the PECO framework enhance the retrieval effectiveness.

Model	**MAP**	**P@5**	**#rel. ret.**
Baseline-1	**0.129**	0.151	5369
Baseline-2	0.128	0.161*	4645
PECO-queries	0.126	**0.172***	**5433**

Table 1: Comparing the performance measures of keyword-based and PECO-structured queries in terms of MAP, precision at 5 and number of relevant documents retrieved (#rel. ret.). ($*$: t.test $<$ 0.05)

In a second series of experiments, we evaluated the model we proposed in Section 5 . We compared two variants of our model. The first variant (named Model-1) uses a global σ_e distribution fixed according to the average distribution of all PECO elements (i.e. the observed probability that a PECO element occurs in a document' part, no matter which element it is). The second variant (named Model-2) uses a differentiated σ_e distribution for each type of PECO element. The idea is to see if, given the fact that PECO elements have different distributions in documents, using an adapted weight distribution for each element can improve the retrieval effectiveness.

Previous studies have shown that assigning a different weight to each PECO element in the query leads to better results (Demner-Fushman and Lin, 2007; Boudin et al., 2010a). In order to compare our model with a similar method, we defined another baseline (named Baseline-3) by fixing the parameters $\beta = 0$ and $\gamma = 0$ in equation 2. We performed a grid search (from 0 to 1 by step of 0.1) to find the optimal δ weights. Regarding the last three parameters in our full models, namely α, β and γ, we conducted a second grid search to find their optimal values. Performance measures obtained in 10-fold cross-validation (optimizing the MAP measure) by these models are presented in Table 2.

A significant improvement is obtained by the Baseline-3 over the keyword-based approach (Baseline-2). The PECO decomposition of queries is particularly useful to balance the importance of each element in the scoring function. We observe a large improvement in retrieval effectiveness for both models over the two baselines. This strongly indicates that a weighting scheme based on the word position in documents is effective. These results support our assumption that the distribution of PECO elements in documents can be used to weight words in the document language model.

However, we do not observe meaningful differences between Model-1 and Model-2. This tend to suggest that a global distribution is likely more robust for IR purposes than separate distributions for each type of element. Another possible reason is that our direct mapping from positional distribution to probabilities may not be the most appropriate. One may think about using a different transformation, or performing some smoothing. We will leave this for our future work.

7 Conclusion

This paper first presented the construction of a test collection for evaluating clinical information retrieval. From a set of systematic reviews, a group of annotators were asked to generate structured clinical queries and collect relevance judgments. The resulting test collection is composed of 423 queries and 8926 relevant documents. This test collection provides a basis for researchers to experiment with PECO-structured queries in clinical IR. The test collection introduced in this paper, along with the manual given to the group of annotators, will be available for download[5].

In a second step, this paper addressed the problem of using the PECO framework in clinical IR. A straightforward idea is to identify PECO elements in documents and use the elements in the retrieval process. However, this approach does not work well be-

[5] http://www-etud.iro.umontreal.ca/~boudinfl/pecodr/

Model	**MAP**	**% rel.**	**P@5**	**% rel.**	**#rel. ret.**
Baseline-2	0.128	-	0.161	-	4645
Baseline-3	0.144	+12.5%*	0.196	+21.7%†	5780
Model-1	0.164	+28.1%†	0.241	+49.7%†	5768
Model-2	0.163	+27.3%†	0.240	+49.1%†	5770

Table 2: 10-fold cross validation scores for the Baseline-2, Baseline-3 and the two variants of our proposed model (Model-1 and Model-2). Relative increase over the Baseline-2 is given, #rel. ret. is the number of relevant documents retrieved. ($\dagger$: t.test < 0.01, $*$: t.test < 0.05)

cause of the difficulty to automatically detect these elements. Instead, we proposed a less demanding approach that uses the distribution of PECO elements in documents to re-weight terms in the document model. The observation of variable distributions in our test collection led us to believe that the position information can be used as a robust indicator of the presence of a PECO element. This strategy turns out to be promising. On a data set composed of 1.5 million citations extracted with PubMed, our best model obtains an increase of 28% for MAP and nearly 50% for P@5 over the classical language modeling approach.

In future work, we intend to expand our analysis of the distribution of PECO elements to a larger number of citations. One way to do that would be to automatically extract PubMed citations that contain structural markers associated to PECO categories (Chung, 2009).

References

Florian Boudin, Jian-Yun Nie, and Martin Dawes. 2010a. Clinical Information Retrieval using Document and PICO Structure. In *Proceedings of the HLT-NAACL 2010 conference*, pages 822–830.

Florian Boudin, Lixin Shi, and Jian-Yun Nie. 2010b. Improving Medical Information Retrieval with PICO Element Detection. In *Proceedings of the ECIR 2010 conference*, pages 50–61.

Grace Y. Chung. 2009. Sentence retrieval for abstracts of randomized controlled trials. *BMC Medical Informatics and Decision Making*, 9(1).

Thomas Owens Sheri Keitz Connie Schardt, Martha B Adams and Paul Fontelo. 2007. Utilization of the PICO framework to improve searching PubMed for clinical questions. *BMC Medical Informatics and Decision Making*, 7(1).

Martin Dawes, Pierre Pluye, Laura Shea, Roland Grad, Arlene Greenberg, and Jian-Yun Nie. 2007. The identification of clinically important elements within medical journal abstracts: PatientPopulationProblem, ExposureIntervention, Comparison, Outcome, Duration and Results (PECODR). *Informatics in Primary care*, 15(1):9–16.

D. Demner-Fushman and J. Lin. 2007. Answering clinical questions with knowledge-based and statistical techniques. *Computational Linguistics*, 33(1):63–103.

William R. Hersh, Katherine Crabtree, David H. Hickam, Lynetta Sacherek, Linda Rose, and Charles P. Friedman. 2000. Factors associated with successful answering of clinical questions using an information retrieval system. *Bulletin of the Medical Library Association*, 88(4):323–331.

Larry McKnight and Padmini Srinivasan. 2003. Categorization of sentence types in medical abstracts. Proceedings of the AMIA annual symposium.

Donald Metzler and W. Bruce Croft. 2005. A Markov random field model for term dependencies. In *Proceedings of the SIGIR conference*, pages 472–479.

Yun Niu, Graeme Hirst, Gregory McArthur, and Patricia Rodriguez-Gianolli. 2003. Answering clinical questions with role identification. In *Proceedings of the ACL 2003 Workshop on Natural Language Processing in Biomedicine*, pages 73–80.

Pierre Pluye, Roland M. Grad, Lynn G. Dunikowski, and Randolph Stephenson. 2005. Impact of clinical information-retrieval technology on physicians: a literature review of quantitative, qualitative and mixed methods studies. *International Journal of Medical Informatics*, 74(9):745–768.

Jay M. Ponte and W. Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of the SIGIR conference*, pages 275–281.

Scott W. Richardson, Mark C. Wilson, Jim Nishikawa, and Robert S. Hayward. 1995. The well-built clinical question: a key to evidence-based decisions. *ACP Journal Club*, 123(3):A12–13.

David L. Sackett, William Rosenberg, J. A. Muir Gray, Brian Haynes, and W. Scott Richardson. 1996. Evidence based medicine: what it is and what it isn't. *British medical journal*, 312:71–72.

Inducing Word Senses to Improve Web Search Result Clustering

Roberto Navigli and **Giuseppe Crisafulli**
Dipartimento di Informatica
Sapienza Università di Roma
navigli@di.uniroma1.it, crisafulli.giu@gmail.com

Abstract

In this paper, we present a novel approach to Web search result clustering based on the automatic discovery of word senses from raw text, a task referred to as Word Sense Induction (WSI). We first acquire the senses (i.e., meanings) of a query by means of a graph-based clustering algorithm that exploits cycles (triangles and squares) in the co-occurrence graph of the query. Then we cluster the search results based on their semantic similarity to the induced word senses. Our experiments, conducted on datasets of ambiguous queries, show that our approach improves search result clustering in terms of both clustering quality and degree of diversification.

1 Introduction

Over recent years increasingly huge amounts of text have been made available on the Web. Popular search engines such as Yahoo! and Google usually do a good job at retrieving a small number of relevant results from such an enormous collection of Web pages (i.e. retrieving with high precision, low recall). However, current search engines are still facing the lexical ambiguity issue (Furnas et al., 1987) – i.e. the linguistic property owing to which any particular word may convey different meanings. In a recent study (Sanderson, 2008) – conducted using WordNet (Miller et al., 1990) and Wikipedia as sources of ambiguous words – it was reported that around 3% of Web queries and 23% of the most frequent queries are ambiguous. Examples include: "buy B-52" (a cocktail? a bomber? a DJ workstation? tickets for a band?), "Alexander Smith quotes" (the novelist? the poet?), "beagle search" (dogs? the Linux search tool? the landing spacecraft?).

Ambiguity is often the consequence of the low number of query words entered on average by Web users (Kamvar and Baluja, 2006). While average query length is increasing – it is now estimated at around 3 words per query[1] – many search engines such as Google have already started to tackle the query ambiguity issue by reranking and diversifying their results, so as to prevent Web pages that are similar to each other from ranking too high on the list.

In the past few years, Web clustering engines (Carpineto et al., 2009) have been proposed as a solution to the lexical ambiguity issue in Web Information Retrieval. These systems group search results, by providing a cluster for each specific aspect (i.e., meaning) of the input query. Users can then select the cluster(s) and the pages therein that best answer their information needs. However, many Web clustering engines group search results on the basis of their lexical similarity. For instance, consider the following snippets returned for the *beagle search* query:

1. *Beagle* is a *search* tool that ransacks your...

2. ...the *beagle* disappearing in *search* of game...

3. *Beagle* indexes your files and *searches*...

While snippets 1 and 3 both concern the Linux search tool, they do not have any content word in

[1] http://www.hitwise.com/us/press-center/press-releases/google-searches-apr-09

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 116–126,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

common except our query words. As a result, they will most likely be assigned to two different clusters.

In this paper we present a novel approach to Web search result clustering which is based on the automatic discovery of word senses from raw text – a task referred to as Word Sense Induction (WSI). At the core of our approach is a graph-based algorithm that exploits cycles in the co-occurrence graph of the input query to detect the query's meanings. Our experiments on two datasets of ambiguous queries show that our WSI approach boosts search result clustering in terms of both clustering quality and degree of diversification.

2 Related Work

Web directories. A first, historical solution to query ambiguity is that of Web directories, that is taxonomies providing categories to which Web pages are manually assigned (e.g., the Open Directory Project – `http://dmoz.org`). Given a query, search results are organized by category. This approach has three main weaknesses: first, it is static, thus it needs manual updates to cover new pages; second, it covers only a small portion of the Web; third, it classifies Web pages based on coarse categories. This latter feature of Web directories makes it difficult to distinguish between instances of the same kind (e.g., pages about artists with the same surname classified as `Arts:Music:Bands and Artists`). While methods for the automatic classification of Web documents have been proposed (e.g., (Liu et al., 2005b; Xue et al., 2008)) and some problems have been effectively tackled (Bennett and Nguyen, 2009), these approaches are usually supervised and still suffer from relying on a predefined taxonomy of categories.

Semantic Information Retrieval (SIR). A different direction consists of associating explicit semantics (i.e., word senses or concepts) with queries and documents, that is, performing Word Sense Disambiguation (WSD, see Navigli (2009)). SIR is performed by indexing and/or searching concepts rather than terms, thus potentially coping with two linguistic phenomena: expressing a single meaning with different words (*synonymy*) and using the same word to express various different meanings (*polysemy*). Over the years, different methods for SIR have been proposed (Krovetz and Croft, 1992; Voorhees, 1993; Mandala et al., 1998; Gonzalo et al., 1999; Kim et al., 2004; Liu et al., 2005a, inter alia). However, contrasting results have been reported on the benefits of these techniques: it has been shown that WSD has to be very accurate to benefit Information Retrieval (Sanderson, 1994) – a result that was later debated (Gonzalo et al., 1999; Stokoe et al., 2003). Also, it has been reported that WSD has to be very precise on minority senses and uncommon terms, rather than on frequent words (Krovetz and Croft, 1992; Sanderson, 2000).

SIR relies on the existence of a reference dictionary to perform WSD (typically, WordNet) and thus suffers from its static nature and its inherent paucity of most proper nouns. This latter problem is particularly important for Web searches, as users tend to retrieve more information about named entities (e.g., singers, artists, cities) than concepts (e.g., abstract information about singers or artists).

Search Result Clustering. A more popular approach to query ambiguity is that of search result clustering. Typically, given a query, the system starts from a flat list of text snippets returned from one or more commonly-available search engines and clusters them on the basis of some notion of textual similarity. At the root of the clustering approach lies van Rijsbergen's (1979) cluster hypothesis: "closely associated documents tend to be relevant to the same requests", whereas documents concerning different meanings of the input query are expected to belong to different clusters.

Approaches to search result clustering can be classified as data-centric or description-centric (Carpineto et al., 2009). The former focus more on the problem of data clustering than on presenting the results to the user. A pioneering example is Scatter/Gather (Cutting et al., 1992), which divides the dataset into a small number of clusters and, after the selection of a group, performs clustering again and proceeds iteratively. Developments of this approach have been proposed which improve on cluster quality and retrieval performance (Ke et al., 2009). Other data-centric approaches use agglomerative hierarchical clustering (e.g., LASSI (Yoelle Maarek and Pelleg, 2000)), rough sets (Ngo and Nguyen, 2005) or exploit link information (Zhang et al., 2008).

Description-centric approaches are, instead, more focused on the description to produce for each cluster of search results. Among the most popular and successful approaches are those based on suffix trees (Zamir et al., 1997; Zamir and Etzioni, 1998), including later developments (Crabtree et al., 2005; Bernardini et al., 2009). Other methods in the literature are based on formal concept analysis (Carpineto and Romano, 2004), singular value decomposition (Osinski and Weiss, 2005), spectral clustering (Cheng et al., 2005), spectral geometry (Liu et al., 2008), link analysis (Gelgi et al., 2007), and graph connectivity measures (Di Giacomo et al., 2007). Search result clustering has also been viewed as a supervised salient phrase ranking task (Zeng et al., 2004).

Diversification. Another recent research topic dealing with the query ambiguity issue is diversification, which aims to rerank top search results based on criteria that maximize their diversity. One of the first examples of diversification algorithms is based on the use of similarity functions to measure the diversity among documents and between document and query (Carbonell and Goldstein, 1998). Other techniques use conditional probabilities to determine which document is most different from higher-ranking ones (Chen and Karger, 2006) or use affinity ranking (Zhang et al., 2005), based on topic variance and coverage. More recently, an algorithm called Essential Pages (Swaminathan et al., 2009) has been proposed to reduce information redundancy and return Web pages that maximize coverage with respect to the input query.

Word Sense Induction (WSI). In contrast to the above approaches, we perform WSI to dynamically acquire an inventory of senses of the input query. Instead of performing clustering on the basis of the surface similarity of Web snippets, we use our induced word senses to group snippets. Very little work on this topic exists: vector-based WSI was successfully shown to improve bag-of-words ad-hoc Information Retrieval (Schütze and Pedersen, 1995) and preliminary studies (Udani et al., 2005; Chen et al., 2008) have provided interesting insights into the use of WSI for Web search result clustering. A more recent attempt at automatically identifying query meanings is based on the use of hidden topics (Nguyen et al., 2009). However, in this approach topics – estimated from a universal dataset – are query-independent and thus their number needs to be established beforehand. In contrast, we aim to cluster snippets based on a dynamic and finer-grained notion of sense.

3 Approach

Web search result clustering is usually performed in three main steps:

1. Given a query q, a search engine (e.g., Yahoo!) is used to retrieve a list of results $R = (r_1, \dots, r_n)$;

2. A clustering $\mathcal{C} = (C_0, C_1, \dots, C_m)$ of the results in R is obtained by means of a clustering algorithm;

3. The clusters in $\mathcal{C}$ are optionally labeled with an appropriate algorithm (e.g., see Zamir and Etzioni (1998) and Carmel et al. (2009)) for visualization purposes.

Our key idea is to improve step 2 by means of a Word Sense Induction algorithm: given a query q, we first dynamically induce, from a text corpus, the set of word senses of q (Section 3.1); next, we cluster the Web results on the basis of the word senses previously induced (Section 3.2).

3.1 Word Sense Induction

Word Sense Induction algorithms are unsupervised techniques aimed at automatically identifying the set of senses denoted by a word. These methods induce word senses from text by clustering word occurrences based on the idea that a given word – used in a specific sense – tends to co-occur with the same neighbouring words (Harris, 1954). Several approaches to WSI have been proposed in the literature (see Navigli (2009) for a survey), ranging from clustering based on context vectors (e.g., Schütze (1998)) to word clustering (e.g., Lin (1998)) and co-occurrence graphs (e.g., Widdows and Dorow (2002)).

Successful approaches such as HyperLex (Véronis, 2004) – a graph algorithm based on the identification of hubs in co-occurrence graphs – have to cope with a high number of parameters to be tuned (Agirre et al., 2006). To deal with this issue we propose two variants of a simple, yet effective, graph-based algorithm for WSI, that we describe hereafter. The algorithm consists of two steps: graph construction and identification of word senses.

3.1.1 Graph construction

Given a target query q, we build a co-occurrence graph $G_q = (V, E)$ such that V is a set of context words related to q and E is the set of undirected edges, each denoting a co-occurrence between pairs of words in V. To determine the set of co-occurring words V, we use the Google Web1T corpus (Brants and Franz, 2006), a large collection of n-grams ($n = 1, \dots, 5$) – i.e., windows of n consecutive tokens – occurring in one terabyte of Web documents. First, for each content word w we collect the total number $c(w)$ of its occurrences and the number of times $c(w, w')$ that w and w' occur together in any 5-gram (we include inflected forms in the count); second, we use the Dice coefficient to determine the strength of co-occurrence between w and w':

$$Dice(w, w') = \frac{2c(w, w')}{c(w) + c(w')}. \qquad (1)$$

The rationale behind Dice is that dividing by the sum of total counts of the two words drastically decreases the ranking of words that tend to co-occur frequently with many other words (e.g., *new*, *old*, *nice*, etc.).

The graph $G_q = (V, E)$ is built as follows:

- Our initial vertex set $V^{(0)}$ contains all the content words from the snippet results of query q (excluding stopwords); then, we add to $V^{(0)}$ the highest-ranking words co-occurring with q in the Web1T corpus, i.e., those words w for which $Dice(q, w) \geq \delta$ (the threshold δ is established experimentally, see Section 4.1). We set $V := V^{(0)}$ and $E := \emptyset$.

- For each word $w \in V^{(0)}$, we select the highest ranking words co-occurring with w in Web1T, that is those words w' for which $Dice(w, w') \geq \delta$. We add each of these words to V (note that some w' might already be in $V^{(0)}$) and the corresponding edge $\{w, w'\}$ to E with weight $Dice(w, w')$. Finally, we remove disconnected vertices.

3.1.2 Identification of word senses

The main idea behind our approach is that edges in the co-occurrence graph participating in cycles are likely to connect vertices (i.e., words) belonging to the same meaning component. Specifically, we focus on cycles of length 3 and 4, called respectively triangles and squares in graph theory.

For each edge e, we calculate the ratio of triangles in which e participates:

$$\mathit{Tri}(e) = \frac{\text{\# triangles } e \text{ participates in}}{\text{\# triangles } e \text{ could participate in}} \qquad (2)$$

where the numerator is the number of cycles of length 3 in which $e = \{w, w'\}$ participates, and the denominator is the total number of neighbours of w and w'. Similarly, we define a measure $Sqr(e)$ of the ratio of squares (i.e., cycles of length 4) an edge e participates in to the number of possible squares e could potentially participate in:

$$\mathit{Sqr}(e) = \frac{\text{\# squares } e \text{ participates in}}{\text{\# squares } e \text{ could participate in}} \qquad (3)$$

where the numerator is the number of squares containing e and the denominator is the number of possible distinct pairs of neighbours of w and w'. If no triangle (or square) exists for e, the value of the corresponding function is set to 0.

In order to disconnect the graph and determine the meaning components, we remove all the edges whose *Tri* (or *Sqr*) value is below a threshold σ. The resulting connected components represent the word senses induced for the query q. Notice that the number of senses is dynamically chosen based on the co-occurrence graph and the algorithm's thresholds.

Our triangular measure is the edge counterpart of the clustering coefficient (or curvature) for vertices, previously used to perform WSI (Widdows and Dorow, 2002). However, it is our hunch that measuring the ratio of squares an edge participates in provides a stronger clue of how important that edge is within a meaning component. In Section 4, we will corroborate this idea with our experiments.

3.1.3 An example

As an example, let $q = \textit{beagle}$. Two steps are performed:

1. **Graph construction.** We build the co-occurrence graph $G_{beagle} = (V, E)$, an excerpt of which is shown in Figure 1(a).

2. **Identification of word senses.** We calculate the *Sqr* values of each edge in the graph. The edges e whose $\mathit{Sqr}(e) < \sigma$ are removed (we assume $\sigma = 0.25$). For instance, *Sqr*({ *dog, breed* }) = $\frac{1}{2}$, as the edge participates in the square *dog – breed – puppy – canine – dog*, but it could also have participated in the potential square *dog – breed – puppy – search – dog*. In fact, the other neighbours of *dog* are *canine*, *puppy* and *search*, and the other neighbour of *breed* is *puppy*, thus the square can only be closed by connecting *puppy* to either *canine* or *search*. In our example, the only edges whose *Sqr* is below σ are: { *dog, puppy* }, { *dog, search* } and { *linux, mission* } (they participate in no square). We remove these edges and select the resulting connected components as the senses of the query *beagle* (shown in Figure 1(b)). Note that, if we selected triangles as our pruning measure, we should also remove the following edges { *search, index* }, { *index, linux* }, { *linux, system* } and { *system, search* }. In fact, these edges do not participate in any triangle (while they do participate in a square). As a result, we would miss the computer science sense of the query.

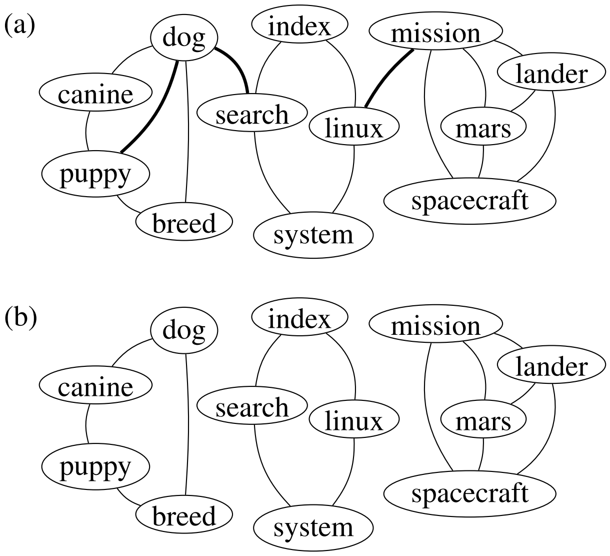


Figure 1: The *beagle* example: (a) graph construction, "weak" edges (according to *Sqr*) drawn in bold, (b) the word senses induced after edge removal.

3.2 Clustering of Web results

Given our query q, we submit it to a search engine, which returns a list of relevant search results $R = (r_1, \dots, r_n)$. We process each result r_i by considering the corresponding text snippet and transforming it to a bag of words b_i (we apply tokenization, stopwords and target word removal, and lemmatization[2]). For instance, given the snippet:

"the *beagle* is a breed of medium-sized dog",

we produce the following bag of words:

{ *breed*, *medium*, *size*, *dog* }.

As a result of the above processing, we obtain a list of bags of words $B = (b_1, \dots, b_n)$. Now, our aim is to cluster our Web results R, i.e., the corresponding bags of words B. To this end, rather than considering the interrelationships between them (as is done in traditional search result clustering), we intersect each bag of words $b_i \in B$ with the sense clusters $\{S_1, \dots, S_m\}$ acquired as a result of our Word Sense Induction algorithm (cf. Section 3.1). The sense cluster with the largest intersection with b_i is selected as the most likely meaning of r_i. Formally:

$$\mathit{Sense}(r_i) = \begin{cases} \underset{j=1,\dots,m}{\operatorname{argmax}} |b_i \cap S_j| & \text{if} \max\limits_j |b_i \cap S_j| > 0 \\ 0 & \text{else} \end{cases} \tag{4}$$

where 0 denotes that no sense is assigned to result r_i, as the intersection is empty for all senses S_j. Otherwise the function returns the index of the sense having the largest overlap with b_i – the bag of words associated with the search result r_i. As a result of sense assignment for each $r_i \in R$, we obtain a clustering $\mathcal{C} = (C_0, C_1, \dots, C_m)$ such that:

$$C_j = \{r_i \in R : \mathit{Sense}(r_i) = j\}, \tag{5}$$

that is, C_j contains the search results classified with the j-th sense of query q (C_0 includes unassigned results). Finally, we sort the clusters in our clustering $\mathcal{C}$ based on their "quality". For each cluster $C_j \in \mathcal{C} \setminus \{C_0\}$, we determine its similarity with

[2] We use the WordNet lemmatizer.

the corresponding meaning S_j by calculating the following formula:

$$avgsim(C_j, S_j) = \frac{\sum_{r_i \in C_j} sim(r_i, S_j)}{|C_j|}. \quad (6)$$

The formula determines the average similarity between the search results in cluster C_j and the corresponding sense cluster S_j. The similarity between a search result r_i and S_j is determined as the normalized overlap between its bag of words b_i and S_j:

$$sim(r_i, S_j) = sim(b_i, S_j) = \frac{|b_i \cap S_j|}{|b_i|}. \quad (7)$$

Finally, we rank the elements r_i within each cluster C_j by their similarity $sim(r_i, S_j)$. We note that the ranking and optimality of clusters can be improved with more sophisticated techniques (Crabtree et al., 2005; Kurland, 2008; Kurland and Domshlak, 2008; Lee et al., 2008, inter alia). However, this is outside the scope of this paper.

4 Experiments

4.1 Experimental Setup

Test Sets. We conducted our experiments on two datasets:

- AMBIENT (AMBIguous ENTries), a recently released dataset which contains 44 ambiguous queries[3]. The sense inventory for the meanings (i.e., subtopics)[4] of queries is given by Wikipedia disambiguation pages. For instance, given the *beagle* query, its disambiguation page in Wikipedia provides the meanings of dog, Mars lander, computer search service, beer brand, etc. The top 100 Web results of each query returned by the Yahoo! search engine were tagged with the most appropriate query senses according to Wikipedia (amounting to 4400 sense-annotated search results). To our knowledge, this is currently the largest dataset of ambiguous queries available on-line. Other datasets, such as those from the TREC competitions, are not focused on distiguishing the subtopics of a query.

dataset	queries	queries by length				avg. polys.
		1	2	3	4	
AMBIENT	44	35	6	3	0	17.9
MORESQUE	114	0	47	36	31	6.7

Table 1: Statistics on the datasets of ambiguous queries.

- MORESQUE (MORE Sense-tagged QUEry results), a new dataset of 114 ambiguous queries which we developed as a complement to AMBIENT following the guidelines provided by its authors. In fact, our aim was to study the behaviour of Web search algorithms on queries of different lengths, ranging from 1 to 4 words. However, the AMBIENT dataset is composed mostly of single-word queries. MORESQUE provides dozens of queries of length 2, 3 and 4, together with the 100 top results from Yahoo! for each query annotated as in the AMBIENT dataset (overall, we tagged 11,400 snippets). We decided to carry on using Yahoo! mainly for homogeneity reasons.

We report the statistics on the composition of the two datasets in Table 1. Given that the snippets could possibly be annotated with more than one Wikipedia subtopic, we also determined the average number of subtopics per snippet. This amounted to 1.01 for AMBIENT and 1.04 for MORESQUE for snippets with at least one subtopic annotation (51% and 53% of the respective datasets). We can thus conclude that multiple subtopic annotations are infrequent.

Parameters. Our graph-based algorithms have two parameters: the Dice threshold δ for graph construction (Section 3.1.1) and the threshold σ for edge removal (Section 3.1.2). The best parameters, used throughout our experiments, were ($\delta = 0.00033, \sigma = 0.45$) with triangles and ($\delta = 0.00033, \sigma = 0.33$) with squares. The parameter values were obtained as a result of tuning on a small in-house development dataset. The dataset was built by automatically identifying monosemous words and creating pseudowords following the scheme proposed by Schütze (1998).

Systems. We compared Triangles and Squares against the best systems reported by Bernardini et al. (2009, cf. Section 2):

[3] `http://credo.fub.it/ambient`

[4] In the following, we use the terms *subtopic* and *word sense* interchangeably.

- **Lingo** (Osinski and Weiss, 2005): a Web clustering engine implemented in the Carrot^2 open-source framework[5] that clusters the most frequent phrases extracted using suffix arrays.

- **Suffix Tree Clustering (STC)** (Zamir and Etzioni, 1998): the original Web search clustering approach based on suffix trees.

- **KeySRC** (Bernardini et al., 2009): a state-of-the-art Web clustering engine built on top of STC with part-of-speech pruning and dynamic selection of the cut-off level of the clustering dendrogram.

- **Essential Pages (EP)** (Swaminathan et al., 2009): a recent diversification algorithm that selects fundamental pages which maximize the amount of information covered for a given query.

- **Yahoo!**: the original search results returned by the Yahoo! search engine.

The first three of the above are Web search result clustering approaches, whereas the last two produce lists of possibly diversified results (cf. Section 2).

4.2 Experiment 1: Clustering Quality

Measure. While assessing the quality of clustering is a notably hard problem, given a gold standard $\mathcal{G}$ we can calculate the **Rand index** (RI) of a clustering $\mathcal{C}$, a common quality measure in the literature, determined as follows (Rand, 1971; Manning et al., 2008):

$$\text{RI}(\mathcal{C}) = \frac{\sum_{(w,w')\in\mathcal{W}\times\mathcal{W},w\neq w'} \delta(w,w')}{|\{(w,w')\in\mathcal{W}\times\mathcal{W} : w \neq w'\}|} \quad (8)$$

where $\mathcal{W}$ is the union set of all the words in $\mathcal{C}$ and $\delta(w,w') = 1$ if any two words w and w' are in the same cluster both in $\mathcal{C}$ and in the gold standard $\mathcal{G}$ or they are in two different clusters in both $\mathcal{C}$ and $\mathcal{G}$, otherwise $\delta(w,w') = 0$. In other words, we calculate the percentage of word pairs that are in the same configuration in both $\mathcal{C}$ and $\mathcal{G}$. For the gold standard $\mathcal{G}$ we use the clustering induced by the sense annotations provided in our datasets for each snippet (i.e., each cluster contains the snippets manually associated with a particular Wikipedia subtopic). Similarly to what was done in Section 3.2, untagged results are grouped together in a special cluster of $\mathcal{G}$.

System	AMBIENT	MORESQUE	All
Squares	72.59	65.41	67.28
Triangles	66.13	64.47	64.93
Lingo	62.75	52.68	55.49
STC	61.48	51.52	54.29
KeySRC	66.49	55.82	58.78

Table 2: Results by Rand index (percentages).

Results. The results of all systems on the AMBIENT and MORESQUE datasets according to the average Rand index are shown in Table 2[6]. In accordance with previous results in the literature, KeySRC performed generally better than the other search result clustering systems, especially on smaller queries. Our Word Sense Induction systems, Squares and Triangles, outperformed all other systems by a large margin, thus showing a higher clustering quality (with the exception of KeySRC performing better than Triangles on AMBIENT). Interestingly, all clustering systems perform more poorly on longer queries (i.e., on the MORESQUE dataset), however our WSI systems, and especially Triangles, are more robust across query lengths. Compared to Triangles, the Squares algorithm performs better, confirming our hunch that Squares is a more solid graph pattern.

4.3 Experiment 2: Diversification

Measure. Search result clustering can also be used to diversify the top results returned by a search engine. Thus, for each query q, one natural way of measuring a system's performance is to calculate the **subtopic recall-at-K** (Zhai et al., 2003) given by the number of different subtopics retrieved for q in the top K results returned:

$$\text{S-recall@K} = \frac{|\bigcup_{i=1}^{K} subtopics(r_i)|}{M} \quad (9)$$

where $subtopics(r_i)$ is the set of subtopics manually assigned to the search result r_i and M is the number of subtopics for query q (note that in our experiments M is the number of subtopics occurring in the 100 results retrieved for q, so S-recall@100 = 1). However, this measure is only suitable for systems returning ranked lists (such as Yahoo! and EP). Given

[5] http://project.carrot2.org

[6] For reference systems we used the implementations of Bernardini et al. (2009) and Osinski and Weiss (2005).

System	K=3	K=5	K=10	K=15	K=20
Squares	51.9	63.4	75.8	83.3	87.4
Triangles	50.8	62.4	75.2	82.7	86.6
Yahoo!	49.2	60.0	72.9	78.5	82.7
EP	40.6	53.2	68.6	77.2	83.3
KeySRC	44.3	55.8	72.0	79.1	83.2

Table 3: S-recall@K on all queries (percentages).

a clustering $\mathcal{C} = (C_0, C_1, \ldots, C_m)$, we flatten it to a list as follows: we add to the initially empty list the first element of each cluster C_j ($j = 1, \ldots, m$); then we iterate the process by selecting the second element of each cluster C_j such that $|C_j| \geq 2$, and so on. The remaining elements returned by the search engine, but not included in any cluster of $\mathcal{C} \setminus \{C_0\}$, are appended to the bottom of the list in their original order. Note that the elements are selected from each cluster according to their internal ranking (e.g., for our algorithms we use Formula 7 introduced in Section 3.2).

Results. For the sake of clarity and to save space, we selected the best systems from our previous experiment, namely Squares, Triangles and KeySRC, and compared their output with the original snippet list returned by Yahoo! and the output of the EP diversification algorithm (cf. Section 4.1).

The S-recall@K (with $K = 3, 5, 10, 15, 20$) calculated on AMBIENT+MORESQUE is reported in Table 3. Squares and Triangles show the highest degree of diversification, with a subtopic recall greater than all other systems, and with Squares consistently performing better than Triangles. It is interesting to observe that KeySRC performs worse than Yahoo! with low values of K and generally better with higher values of K.

Given that the two datasets complement each other in terms of query lengths (with AMBIENT having queries of length $\leq$ 2 and MORESQUE with many queries of length $\geq$ 3), we studied the S-recall@K trend for the two datasets. The results are shown in Figures 2 and 3. While KeySRC does not show large differences in the presence of short and long ambiguous queries, our graph-based algorithms do. For instance, as soon as $K = 3$ the Squares algorithm obtains S-recall values of 37% and 57.5% on AMBIENT and MORESQUE, respectively. The difference decreases as K increases, but is still significant when K = 10. We hypothesize that, because they are less ambiguous, longer queries are easier to diversify with the aid of WSI. However, we note that, even with low values of K, Squares and Triangles obtain higher S-recall than the other systems (with KeySRC competing on AMBIENT when $K \leq$ 15). Finally, we observe that – with low values of K – the Squares algorithm performs significantly better than Triangles on shorter queries, and only slightly better on longer ones.

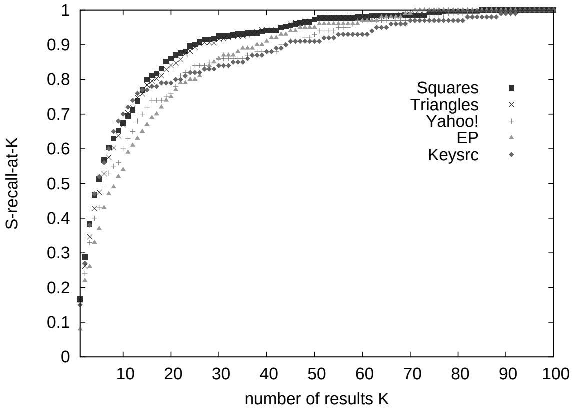


Figure 2: Results by S-recall@K on AMBIENT.

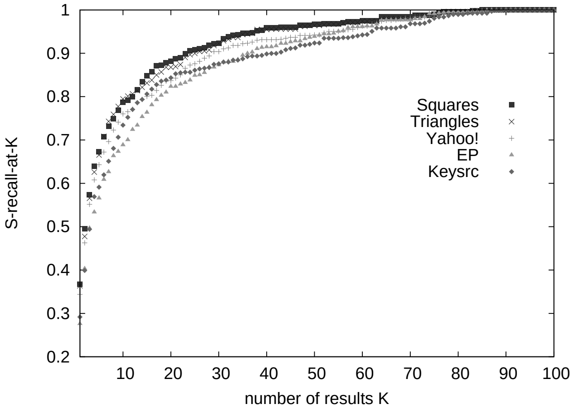


Figure 3: S-recall@K on MORESQUE.

5 Discussion

Results. Our results show that our graph-based algorithms are able to consistently produce clusters of better quality than all other systems tested in our experiments. The results on S-recall@K show that our approach can also be used effectively as a diversification technique, performing better than a very recent proposal such as Essential Pages. The latter outperforms Yahoo! and KeySRC when $K \geq 30$ on AMBIENT, whereas on MORESQUE it performs generally worse until higher values of K are reached. If we analyze the entire dataset of 158 queries by length, EP works best after examining at least 20 results on 1- and 2-word ambiguous queries, whereas on longer queries a larger number of documents (≥ 30) needs to be analyzed before surpassing Yahoo! performance.

The above considerations might not seem intuitive at first glance, as the average polysemy of longer queries is lower (17.9 on AMBIENT vs. 6.7 on MORESQUE according to our gold standard). However, we note that while the kind of ambiguity of 1-word queries is generally coarser (e.g., *beagle* as dog vs. lander vs. search tool), with longer queries we often encounter much finer sense distinctions (e.g., *Across the Universe* as song by The Beatles vs. a 2007 film based on the song vs. a Star Trek novel vs. a rock album by Trip Shakespeare, etc.). Word Sense Induction is able to deal better with this latter kind of ambiguity as discriminative words become part of the meanings acquired.

Performance issues. Inducing word senses from the query graph comes at a higher computational cost than other non-semantic clustering techniques. Indeed, the most time-consuming phase of our approach is the construction of the query graph, which requires intensive querying of our database of co-occurrences calculated from the Web1T corpus. While graphs can be precomputed or cached, previously unseen queries will still require the construction of new graphs. Instead, triangles and squares, as well as the resulting connected components, can be calculated on the fly.

6 Conclusions

In this paper we have presented a novel approach to Web search result clustering. Our key idea is to induce senses for the target query automatically by means of a graph-based algorithm focused on the notion of cycles. The results of a Web search engine are then mapped to the query senses and clustered accordingly.

The paper provides three novel contributions. First, we show that WSI boosts the quality of search result clustering and improves the diversification of the snippets returned as a flat list. We provide a clear indication on the usefulness of a loose notion of sense to cope with ambiguous queries. This is in contrast to research on Semantic Information Retrieval, which has obtained contradictory and often inconclusive results. The main advantage of WSI lies in its dynamic production of word senses that cover both concepts (e.g., *beagle* as a breed of dog) and instances (e.g., *beagle* as a specific instance of a space lander). In contrast, static dictionaries such as WordNet – typically used in Word Sense Disambiguation – by their very nature encode mainly concepts. Second, we propose two simple, yet effective, graph algorithms to induce the senses of our queries. The best performing approach is based on squares (cycles of length 4), a novel graph pattern in WSI. Third, we contribute a new dataset of 114 ambiguous queries and 11,400 sense-annotated snippets which complements an existing dataset of ambiguous queries[7]. Given the lack of ambiguous query datasets available (Sanderson, 2008), we hope our new dataset will be useful in future comparative experiments. Finally, we note that our approach needed very little tuning. Moreover, its requirement of a Web corpus of n-grams is not a stringent one, as such corpora are available for several languages and can be produced for any language of interest.

As regards future work, we intend to combine our clustering algorithm with a cluster labeling algorithm. We also aim to implement a number of Word Sense Induction algorithms and compare them in the same evaluation framework with more Web search and Web clustering engines. Finally, it should be possible to use precisely the same approach presented in this paper for document clustering, by grouping the contexts in which the target query occurs – and we will also experiment on this in the future.

Acknowledgments

We thank Google for providing the Web1T corpus for research purposes. We also thank Massimiliano D'Amico for producing the output of KeySRC and EP, and Stanislaw Osinski and Dawid Weiss for their

[7] The MORESQUE dataset is available at the following URL: `http://lcl.uniroma1.it/moresque`

help with Lingo and STC. Additional thanks go to Jim McManus, Senja Pollak and the anonymous reviewers for their useful comments.

References

Eneko Agirre, David Martínez, Oier López de Lacalle, and Aitor Soroa. 2006. Evaluating and optimizing the parameters of an unsupervised graph-based WSD algorithm. In *Proc. of TextGraphs '06*, pages 89–96, New York, USA.

Paul N. Bennett and Nam Nguyen. 2009. Refined experts: improving classification in large taxonomies. In *Proc. of SIGIR '09*, pages 11–18, Boston, MA, USA.

Andrea Bernardini, Claudio Carpineto, and Massimiliano D'Amico. 2009. Full-subtopic retrieval with keyphrase-based search results clustering. In *Proc. of WI '09*, pages 206–213, Milan, Italy.

Thorsten Brants and Alex Franz. 2006. Web 1t 5-gram, ver. 1, ldc2006t13. In *LDC*, PA, USA.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proc. of SIGIR '98*, pages 335–336, Melbourne, Australia.

David Carmel, Haggai Roitman, and Naama Zwerdling. 2009. Enhancing cluster labeling using Wikipedia. In *Proc. of SIGIR '09*, pages 139–146, MA, USA.

Claudio Carpineto and Giovanni Romano. 2004. Exploiting the potential of concept lattices for information retrieval with CREDO. *Journal of Universal Computer Science*, 10(8):985–1013.

Claudio Carpineto, Stanislaw Osiński, Giovanni Romano, and Dawid Weiss. 2009. A survey of web clustering engines. *ACM Computing Surveys*, 41(3):1–38.

Harr Chen and David R. Karger. 2006. Less is more: probabilistic models for retrieving fewer relevant documents. In *Proc. of SIGIR '06*, pages 429–436, Seattle, WA, USA.

Jiyang Chen, Osmar R. Zaïane, and Randy Goebel. 2008. An unsupervised approach to cluster web search results based on word sense communities. In *Proc. of WI-IAT 2008*, pages 725–729, Sydney, Australia.

David Cheng, Santosh Vempala, Ravi Kannan, and Grant Wang. 2005. A divide-and-merge methodology for clustering. In *Proc. of PODS '05*, pages 196–205, New York, NY, USA.

Daniel Crabtree, Xiaoying Gao, and Peter Andreae. 2005. Improving web clustering by cluster selection. In *Proc. of WI '05*, pages 172–178, Compiègne, France.

Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. 1992. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proc. of SIGIR '92*, pages 318–329, Copenhagen, Denmark.

Emilio Di Giacomo, Walter Didimo, Luca Grilli, and Giuseppe Liotta. 2007. Graph visualization techniques for web clustering engines. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):294–304.

George W. Furnas, Thomas K. Landauer, Louis Gomez, and Susan Dumais. 1987. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):964–971.

Fatih Gelgi, Hasan Davulcu, and Srinivas Vadrevu. 2007. Term ranking for clustering web search results. In *Proc. of WebDB '07*, Beijing, China.

Julio Gonzalo, Anselmo Penas, and Felisa Verdejo. 1999. Lexical ambiguity and Information Retrieval revisited. In *Proc. of EMNLP/VLC 1999*, pages 195–202, College Park, MD, USA.

Zellig Harris. 1954. Distributional structure. *Word*, 10:146–162.

Maryam Kamvar and Shumeet Baluja. 2006. A large scale study of wireless search behavior: Google mobile search. In *Proc. of CHI '06*, pages 701–709, New York, NY, USA.

Weimao Ke, Cassidy R. Sugimoto, and Javed Mostafa. 2009. Dynamicity vs. effectiveness: studying online clustering for scatter/gather. In *Proc. of SIGIR '09*, pages 19–26, MA, USA.

Sang-Bum Kim, Hee-Cheol Seo, and Hae-Chang Rim. 2004. Information Retrieval using word senses: root sense tagging approach. In *Proc. of SIGIR '04*, pages 258–265, Sheffield, UK.

Robert Krovetz and William B. Croft. 1992. Lexical ambiguity and Information Retrieval. *ACM Transactions on Information Systems*, 10(2):115–141.

Oren Kurland and Carmel Domshlak. 2008. A rank-aggregation approach to searching for optimal query-specific clusters. In *Proc. of SIGIR '08*, pages 547–554, Singapore.

Oren Kurland. 2008. The opposite of smoothing: a language model approach to ranking query-specific document clusters. In *Proc. of SIGIR '08*, pages 171–178, Singapore.

Kyung Soon Lee, W. Bruce Croft, and James Allan. 2008. A cluster-based resampling method for pseudo-relevance feedback. In *Proc. of SIGIR '08*, pages 235–242, Singapore.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proc. of the* 17^{th} *COLING*, pages 768–774, Montreal, Canada.

Shuang Liu, Clement Yu, and Weiyi Meng. 2005a. Word Sense Disambiguation in queries. In *Proc. of CIKM '05*, pages 525–532, Bremen, Germany.

Tie-Yan Liu, Yiming Yang, Hao Wan, Hua-Jun Zeng, Zheng Chen, and Wei-Ying Ma. 2005b. Support vector machines classification with a very large-scale taxonomy. *SIGKDD Explor. Newsl.*, 7(1):36–43.

Ying Liu, Wenyuan Li, Yongjing Lin, and Liping Jing. 2008. Spectral geometry for simultaneously clustering and ranking query search results. In *Proc. of SIGIR '08*, pages 539–546, Singapore.

Rila Mandala, Takenobu Tokunaga, and Hozumi Tanaka. 1998. The use of WordNet in Information Retrieval. In *Proc. of the COLING-ACL workshop on Usage of Wordnet in Natural Language Processing*, pages 31–37, Montreal, Canada.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

George A. Miller, Richard T. Beckwith, Christiane D. Fellbaum, Derek Gross, and Katherine Miller. 1990. WordNet: an online lexical database. *International Journal of Lexicography*, 3(4):235–244.

Roberto Navigli. 2009. Word Sense Disambiguation: a survey. *ACM Computing Surveys*, 41(2):1–69.

Chi Lang Ngo and Hung Son Nguyen. 2005. A method of web search result clustering based on rough sets. In *Proc. of WI '05*, pages 673–679, Compiègne, France.

Cam-Tu Nguyen, Xuan-Hieu Phan, Susumu Horiguchi, Thu-Trang Nguyen, and Quang-Thuy Ha. 2009. Web search clustering and labeling with hidden topics. *ACM Transactions on Asian Language Information Processing*, 8(3):1–40.

Stanislaw Osinski and Dawid Weiss. 2005. A concept-driven algorithm for clustering search results. *IEEE Intelligent Systems*, 20(3):48–54.

William M. Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850.

Mark Sanderson. 1994. Word Sense Disambiguation and Information Retrieval. In *Proc. of SIGIR '94*, pages 142–151, Dublin, Ireland.

Mark Sanderson. 2000. Retrieving with good sense. *Information Retrieval*, 2(1):49–69.

Mark Sanderson. 2008. Ambiguous queries: test collections need more sense. In *Proc. of SIGIR '08*, pages 499–506, Singapore.

Hinrich Schütze and Jan Pedersen. 1995. Information Retrieval based on word senses. In *Proceedings of SDAIR'95*, pages 161–175, Las Vegas, Nevada, USA.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–124.

Christopher Stokoe, Michael J. Oakes, and John I. Tait. 2003. Word Sense Disambiguation in Information Retrieval revisited. In *Proc. of SIGIR '03*, pages 159–166, Canada.

Ashwin Swaminathan, Cherian V. Mathew, and Darko Kirovski. 2009. Essential pages. In *Proc. of WI '09*, pages 173–182, Milan, Italy.

Goldee Udani, Shachi Dave, Anthony Davis, and Tim Sibley. 2005. Noun sense induction using web search results. In *Proc. of SIGIR '05*, pages 657–658, Salvador, Brazil.

Cornelis Joost van Rijsbergen. 1979. *Information Retrieval*. Butterworths, second edition.

Jean Véronis. 2004. HyperLex: lexical cartography for Information Retrieval. *Computer Speech and Language*, 18(3):223–252.

Ellen M. Voorhees. 1993. Using WordNet to disambiguate word senses for text retrieval. In *Proc. of SIGIR '93*, pages 171–180, Pittsburgh, PA, USA.

Dominic Widdows and Beate Dorow. 2002. A graph model for unsupervised lexical acquisition. In *Proc. of the* 19^{th} *COLING*, pages 1–7, Taipei, Taiwan.

Gui-Rong Xue, Dikan Xing, Qiang Yang, and Yong Yu. 2008. Deep classification in large-scale text hierarchies. In *Proc. of SIGIR '08*, pages 619–626, Singapore.

Israel Ben-Shaul Yoelle Maarek, Ron Fagin and Dan Pelleg. 2000. Ephemeral document clustering for web applications. *IBM Research Report RJ 10186*.

Oren Zamir and Oren Etzioni. 1998. Web document clustering: a feasibility demonstration. In *Proc. of SIGIR '98*, pages 46–54, Melbourne, Australia.

Oren Zamir, Oren Etzioni, Omid Madani, and Richard M. Karp. 1997. Fast and intuitive clustering of web documents. In *Proc. of KDD '97*, pages 287–290, Newport Beach, California.

Hua-Jun Zeng, Qi-Cai He, Zheng Chen, Wei-Ying Ma, and Jinwen Ma. 2004. Learning to cluster web search results. In *Proc. of SIGIR '04*, pages 210–217, Sheffield, UK.

ChengXiang Zhai, William W. Cohen, and John Lafferty. 2003. Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval. In *Proc. of SIGIR '03*, pages 10–17, Toronto, Canada.

Benyu Zhang, Hua Li, Yi Liu, Lei Ji, Wensi Xi, Weiguo Fan, Zheng Chen, and Wei-Ying Ma. 2005. Improving web search results using affinity graph. In *Proc. of SIGIR '05*, pages 504–511, Salvador, Brazil.

Xiaodan Zhang, Xiaohua Hu, and Xiaohua Zhou. 2008. A comparative evaluation of different link types on enhancing document clustering. In *Proc. of SIGIR '08*, pages 555–562, Singapore.

Improving Translation via Targeted Paraphrasing

Philip Resnik
Linguistics and UMIACS
University of Maryland
resnik@umd.edu

Olivia Buzek
Linguistics and Computer Science
University of Maryland
olivia.buzek@gmail.com

Chang Hu
Computer Science
University of Maryland
changhu@cs.umd.edu

Yakov Kronrod
Linguistics and UMIACS
University of Maryland
yakov@umd.edu

Alex Quinn
Computer Science
University of Maryland
aq@cs.umd.edu

Benjamin B. Bederson
Computer Science and UMIACS
University of Maryland
bederson@cs.umd.edu

Abstract

Targeted paraphrasing is a new approach to the problem of obtaining cost-effective, reasonable quality translation that makes use of simple and inexpensive human computations by monolingual speakers in combination with machine translation. The key insight behind the process is that it is possible to spot likely translation errors with only monolingual knowledge of the target language, and it is possible to generate alternative ways to say the same thing (i.e. paraphrases) with only monolingual knowledge of the source language. Evaluations demonstrate that this approach can yield substantial improvements in translation quality.

1 Introduction

For most of the world's languages, the availability of translation is limited to two possibilities: high quality at high cost, via professional translators, and low quality at low cost, via machine translation (MT). The spectrum between these two extremes is very poorly populated, and at any point on the spectrum the ready availability of translation is limited to only a small fraction of the world's languages. There is, of course, a long history of technological assistance to translators, improving cost effectiveness using translation memory (Laurian, 1984; Bowker and Barlow, 2004) or other interactive tools to assist translators (Esteban et al., 2004; Khadivi et al., 2006). And there is a recent and rapidly growing interest in crowdsourcing with non-professional translators, which can be remarkably effective (Munro, 2010). However, all these alternatives face a central availability bottleneck: they require the participation of humans with bilingual expertise.

In this paper, we report on a new exploration of the middle ground, taking advantage of a virtually unutilized resource: speakers of the source and target language who are *effectively monolingual*, i.e. who each only know one of the two languages relevant for the translation task. The solution we are proposing has the potential to provide a more cost effective approach to translation in scenarios where machine translation *would* be considered acceptable to use, if only it were generally of high enough quality. This would clearly exclude tasks like translation of medical reports, business contracts, or literary works, where the validation of a qualified bilingual translator is absolutely necessary. However, it does include a great many real-world scenarios, such as following news reports in another country, reading international comments about a product, or generating a decent first draft translation of a Wikipedia page for Wikipedia editors to improve.

The use of monolingual participants in a human-machine translation process is not entirely new. Callison-Burch et al. (2004) pioneered the exploration of monolingual post-editing within the MT community, an approach extended more recently to provide richer information to the user by Albrecht et al. (2009) and Koehn (2009). There have also been at least two independently developed human-machine translation frameworks that employ an iterative protocol involving monolinguals on both the source and target side. Morita and Ishida (2009) describe a system in which target and source language speakers perform editing of MT output to improve fluency and adequacy, respectively; they utilize source-side paraphrasing at a course grain level, although their approach is limited to requests to paraphrase the entire sentence when the translation cannot be understood.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 127–137,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

Bederson et al. (2010) describe a similar protocol in which cross-language communication is enhanced by metalinguistic communication in the user interface. Shahaf and Horvitz (2010) use machine translation as a specific instance of a general game-based framework for combining a range of machine and human capabilities.

We call the technique used here *targeted paraphrasing*. In a nutshell, target-language monolinguals identify parts of an initial machine translation that don't appear to be right, and source-language monolinguals provide the MT system with alternative phrasings that might lead to better translations; these are then passed through MT again and the best scoring hypothesis is selected as the final translation. This technique can be viewed as compatible with the richer protocol- and game-based approaches, but it is considerably simpler; in Sections 2 through 4 we describe the method and present evaluation results on Chinese-English translation. Unlike other approaches, the technique also offers clear opportunities to replace human participation with machine components if the latter are up to the task; we discuss this in Section 5 before wrapping up in Section 6 with conclusions and directions for future work.

2 Targeted Paraphrasing

The starting point for our approach is an observation: the source sentence provided as input to an MT system is just one of many ways in which the meaning could have been expressed, and for any given MT system, some forms of expression are easier to translate than others. The same basic observation has been applied quite fruitfully over the past several years to deal with statistical MT challenges involving segmentation, morphological analysis, and more recently, source language word order (Dyer, 2007; Dyer et al., 2008; Dyer and Resnik, 2010). Here we apply it to the surface expression of meaning.

For example, consider the following real example of translation from English to French by an automatic MT system:

- **Source:** Polls indicate Brown, a state senator, and Coakley, Massachusetts' Attorney General, are locked in a virtual tie to fill the late Sen. Ted Kennedy's Senate seat.

- **System:** Les sondages indiquent Brown, un sénateur d'état, et Coakley, Massachusetts' Procureur général, sont enfermés dans une cravate virtuel à remplir le regretté sénateur Ted Kennedy's siège au Sénat.

A French speaker can look at this automatic translation and see immediately that the underlined parts are wrong, even without knowing the intended source meaning. We can identify the spans in the source English sentence that are responsible for these badly translated French spans, and change them to alternative expressions with the same meaning (e.g. changing *Massachusetts' Attorney General* to *the Attorney General of Massachusetts*); if we do so and then use the same MT system again, we obtain a translation that is still imperfect (e.g. *cravate* means necktie), but is more acceptable:

- **System:** Les sondages indiquent que Brown, un sénateur d'état, et Coakley, le procureur général du Massachusetts, sont enfermés dans une cravate virtuel pourvoir le siége au Sénat de Sen. Ted Kennedy, qui est décédé récemment.

Operationally, then, translation with targeted paraphrasing includes the following steps.

Initial machine translation. For this paper, we use the Google Translate Research API, which, among other advantages, provides word-level alignments between the source text and its output. In principle, however, any automatic translation system can be used in this role, potentially at some cost to quality, by performing *post hoc* target-to-source alignment.

Identification of mistranslated spans. This step identifies parts of the source sentence that lead to ungrammatical, nonsensical, or apparently incorrect translations on the target side. In the experiments of Sections 3 and 4, this step is performed by having monolingual target speakers identify likely error spans on the target side, as in the French example above, and projecting those spans back to the source spans that generated them using word alignments as the bridge (Hwa et al., 2005; Yarowsky et al., 2001). In Section 5, we describe a heuristic but effective method for performing this fully automatically. Du et al. (2010), in this proceedings, explore the use of source paraphrases *without* targeting apparent mistranslations, using lattice translation (Dyer et al., 2008) to efficiently represent and decode the resulting very large space of paraphrase alternatives.

Source paraphrase generation. This step generates alternative expressions for the source spans identified in the previous step. In this paper, it is performed by monolingual source speakers who perform the paraphrase task: the speaker is given a sentence with a phrase span marked, and is asked to replace the marked text with a different way of saying the same thing, so that the resulting sentence still makes sense and means the same thing as the original sentence. To illustrate in English, someone seeing *John and Mary took a European vacation this summer* might supply the paraphrase *Mary went on a European*, verifying that the resulting *John and Mary went on a European vacation this summer* preserves the original meaning. This step can also be fully automated (Max, 2009) by taking advantage of bilingual phrase-table pivoting (Bannard and Callison-Burch, 2005); see Max (2010), in these proceedings, for a related approach in which the paraphrases of a source phrase are used to refine the estimated probability distribution over its possible target phrases.

Generating sentential source paraphrases. For each sentence, there may be multiple paraphrased spans. These are multiplied out to provide full-sentence paraphrases. For example, if two non-overlapping source spans are each paraphrased in three ways, we generate 9 sentential source paraphrases, each of which represents an alternative way of expressing the original sentence.

Machine translation of alternative sentences. The alternative source sentences, produced via paraphrase, are sent through the same MT system, and a single-best translation hypothesis is selected, e.g. on the basis of the translation system's model score. In principle, one could also combine the alternatives into a lattice representation and decode to find the best path using lattice translation (Dyer et al., 2008); cf. Du et al. (2010). One could also present translation alternatives to a target speaker for selection, similarly to Callison-Burch et al. (2004).

Notice that with the exception of the initial translation, each remaining step in this pipeline can involve either human participation or fully automatic processing. The targeted paraphrasing framework therefore defines a rich set of intermediate points on the spectrum between fully automatic and fully human translation, of which we explore only a few in this paper.

3 Pilot Study

In order to assess the potential of our approach, we conducted a small pilot study, using eleven sentences in simplified Chinese selected from the article on "Water" in Chinese Wikipedia (http://zh.wikipedia.org/zh-cn/%E6%B0%B4). This article was chosen because its topic is well known in both English-speaking and Chinese-speaking populations. The first five sentences were taken from the first paragraph of the article. The other six sentences were taken from a randomly-chosen paragraph in the article. As a preprocessing step, we removed any parenthetical items from the input sentences, e.g. "(H$_2$0)". The shortest sentence in this set has 12 Chinese characters, the longest has 54.[1]

Human participation in this task was accomplished using Amazon Mechanical Turk, an online marketplace that enables human performance of small "human intelligence tasks" (HITs) in return for micropayments. For each sentence, after we translated it automatically (using Google Translate), three English-speaking Mechanical Turk workers ("Turkers") on the target side performed identification of mistranslated spans. Each span identified was projected back to its corresponding source span, and three Chinese-speaking Turkers were asked to provide paraphrases of each source span. These tasks were easy to perform (no more than around 30 seconds to complete on average) and inexpensive (less than $1 for the entire pilot study).[2] The Chinese source span paraphrases were then used to construct full-sentence paraphrases, which were retranslated, once again by Google Translate, to produce the output of the targeted paraphrasing translation process.

[1] Note that this page is *not* a translation of the corresponding English Wikipedia page or vice versa.

[2] The four English-speaking Turkers were recruited through the normal Mechanical Turk mechanism. The three Chinese-speaking Turkers were recruited offline by the authors in order to quickly obtain results, although they participated as full-fledged Turkers.

The initial translation outputs from Google Translate (GT) and the results of the targeted paraphrasing translation process (TP) were evaluated according to widely used critera of fluency and adequacy. Fluency ratings were obtained on a 5-point scale from three native English speakers without knowledge of Chinese. Translation adequacy ratings were obtained from three native Chinese speakers who are also fluent in English; they assessed adequacy of English sentences by comparing the communicated meaning to the Chinese source sentences.

Fluency was rated on the following scale:

1. Unintelligible: nothing or almost nothing of the sentence is comprehensible.
2. Barely intelligible: only a part of the sentence (less than 50%) is understandable.
3. Fairly intelligible: the major part of the sentence passes.
4. Intelligible: all the content of the sentence is comprehensible, but there are errors of style and/or of spelling, or certain words are missing.
5. Very intelligible: all the content of the sentence is comprehensible. There are no mistakes.

Adequacy was rated on the following scale:

1. None of the meaning expressed in the reference sentence is expressed in the sentence.
2. Little of the reference sentence meaning is expressed in the sentence.
3. Much of the reference sentence meaning is expressed in the sentence.
4. Most of the reference sentence meaning is expressed in the sentence.
5. All meaning expressed in the reference sentence appears in the sentence.

For each GT output, we averaged across the ratings of the alternative TP to produce average TP fluency and adequacy scores. The average GT output ratings, measuring the pure machine translation baseline, were 2.36 for fluency and 2.91 for adequacy. Averaging across the TP outputs, these rose to 3.32 and 3.49, respectively.

One could argue that a more sensible evaluation is not to *average* across alternative TP outputs, but rather to simulate the behavior of a target-language speaker who simply chooses the one translation among the alternatives that seems most fluent. If we select the most fluent TP output for each source sentence according to the English-speakers' average fluency ratings, we obtain average test set ratings of 3.58 for fluency and 3.73 for adequacy. Those are respective gains of 0.82 and 1.21 over the baseline initial MT output, each on a 5-point scale.

Figure 1 shows a selection of outputs: we present the two cases where the most fluent TP alternative shows the greatest gain in average fluency rating (best gain +2.67); two cases near the median gain in average fluency (median +1); and the worst two cases with respect to effect on average fluency rating (worst -0.33). The table accurately conveys a qualitative impression corresponding to the quantitative results: the overall quality of translations appears to be improved by our process consistently, despite the absence of any bilingual input in the improvements.

4 Chinese-English Evaluation

As a followup to our pilot study, we conducted an evaluation using Chinese-English test data taken from the NIST MT'08 machine translation evaluation, in order to obtain fully automatic translation evaluation scores. We report on results for 49 sentences of the 1,357 in this data set. These underwent the same targeted paraphrasing process as in the pilot study, with the addition of a basic step to filter out cheaters: we disregarded as invalid any responses consisting purely of ASCII characters (signifying a non-Chinese response) or responses that were identical to the original source text.

Target English speakers identified 115 potential mistranslation spans, or 2.3 spans per sentence, that yielded at least one source paraphrase on the source Chinese side. Chinese speakers provided 138 valid paraphrases. The entire cost for the human tasks in this experiment was $5.06, or a bit under $0.11 per sentence on average.[3]

Table 1 reports on the results, evaluating in standard fashion using BLEU with the four English MT'08 references for each Chinese sentence. Since the targeted paraphrasing translation process (TP) produces multiple hypotheses — one automatic translation output per sentential paraphrases — we selected the single best output for each sentence by

[3] Invalid paraphrase responses were rejected, i.e. zero-cost.

Condition	Fluency	Adequacy	Sentence
GT	1.33	2.33	Water play life evolve into important to use.
TP	4.00	4.33	Water in the evolution of life played an important role.
GT	1.33	2.67	Human civilization from the source of the majority of large rivers in the domain.
TP	3.33	4.67	Most of the origin of human civilization in river basin.
GT	2.33	3.00	In human daily life, the water in drinking, cleaning, washing and other side to make use of an indispensable.
TP	3.67	3.33	In human daily life, water for drinking, cleaning, washing and other essential role.
GT	2.00	2.33	Eastern and Western ancient Pak prime material view of both the water regarded as a kind of basic groups into the elements, water is the Chinese ancient five rows of a; the West ancient four elements that also have water.
TP	3.00	3.33	East and West in ancient concept of simple substances regarded water as a basic component elements. Among them, the five elements of water is one of ancient China; Western ancient four elements that also have water.
GT	4.00	4.00	Early cities will generally be in the water side of the establishment, in order to solve irrigation, drinking and sewage problems.
TP	4.67	4.33	Early cities are generally built near the water to solve the irrigation, drinking and sewage problems.
GT	3.0	3.33	Human very early on began to produce a water awareness.
TP	2.67	3.00	Man long ago began to understand the water produced.

Figure 1: Original Google Translate output (GT) for the pilot study in Section 3, together with translations produced by the targeted paraphrase translation process (TP), selected to show a range from strong to weak improvements in fluency.

Condition	BLEU
GT (baseline)	28.33
GT n-best oracle	28.47
TP one-best	30.01
TP oracle	30.79
Human upper bound	49.41

Table 1: Results on a 49-sentence subset of the NIST MT'08 Chinese-English test set

selecting the highest scoring English translation, according to the translation score delivered with each output by the Google Translate Research API. (The original translation was, of course, included among the candidates for selection.) This yielded an improvement of 1.68 BLEU points on the 49-sentence test set (TP one-best).

One could argue that this result is simply a result of having more hypotheses to choose from, not a result of the targeted paraphrasing process itself. In order to rule out this possibility, we generated $(n+1)$-best Google translations, setting n for each sentence to match the number of alternative translations generated via targeted paraphrasing. We then chose the best translation for each sentence, among the $(n+1)$-best Google hypotheses, via oracle selection, using the TERp metric (Snover et al., 2009) to evaluate each hypothesis against the reference translations.[4] The resulting BLEU score for the full set showed negligible improvement (GT n-best oracle).

We did a similar oracle-best calculation using TERp for targeted paraphrasing (TP oracle). The result shows a potential gain of 2.46 BLEU points over the baseline, if the best scoring alternative from the targeted paraphrasing process were always chosen.

In addition to aggregate scoring using BLEU, we also looked at oracle results on a per-sentence basis using TERp (since BLEU more appropriate to use at the document level, not the sentence level). Identifying the best sentential paraphrase alternative using TERp as an oracle, we find that the TERp score would improve for 32 of the 49 test sentences, 65.3%. For those 32 sentences, the average gain is 8.36 TERp points.[5] A fairer measure is the average obtained when scoring zero gain for the 17 sentences where no improvement was obtained; taking these into account, i.e. assuming an oracle who chooses the original translation if none of the paraphrase-based alternatives are better, the average improvement over the entire set of 49 sentences is 5.46 TERp points.

Although we have obtained results on only a small subset of the full NIST MT'08 test set, our automatic evaluation confirms the qualitative impressions in Figure 1 and the subjective ratings results obtained in our pilot study in Section 3. The TP oracle results establish that by taking advantage of monolingual human speakers, it is possible to obtain quite substantial gains in translation quality. The TP one-best results demonstrate that the majority of that oracle gain is obtained in automatic hypothesis selection, simply by selecting the paraphrase-based alternative translation with the highest translation score.

The last line in Table 1 shows a human upper bound computed using the reference translations via cross validation; that is, for each of the four reference translations, we evaluate it as a hypothesized translation using the other three references as ground truth; these four scores were then averaged. The value of this upper bound is quite consistent with the bound computed similarly by Callison-Burch (2009).

5 English-Chinese Evaluation

As we noted in Section 2, the targeted paraphrasing translation process defines a set of human-machine combinations that do not require bilingual expertise. The previous section described human identification of mistranslated spans on the target side, human generation of paraphrases for problematic sub-sentential spans on the source side, and both automatic hypothesis selection and human selection (via fluency ratings, in Section 3).

In this section, we take a step toward more automated processing, replacing human identification of mistranslated spans with an a fully automatic method.[6] The idea behind our automatic error identification is straightforward: if the source sentence

[4] An "oracle" telling us which variant is best is not available in the real world, of course, but in situations like this one, oracle studies are often used to establish the magnitude of the potential gain (Och et al., 2004).

[5] "Gains" refer to a lower score: since TERp is an error measure, lower is better.

[6] This section contains material we originally reported in Buzek et al. (2010).

GT: WTO chief negotiator on behalf of the United States to propose substantial reduction of agricultural subsidies, Kai Fa countries substantially reduce industrial products import tariffs to Dapo ?? Doha Round of negotiations deadlock.

TP: World Trade Organization negotiator suggested the United States today, a substantial reduction of agricultural subsidies, developing countries substantially reduce industrial products?? Import tariffs, in order to break the deadlock in the Doha Round of trade negotiations.

REF: the main delegates at the world trade organization talks today suggested that the us make major cuts in its agricultural subsidies and that developing countries significantly reduce import duties on industrial products in order to break the deadlock in the doha round of trade talks .

GT: Emergency session of the Palestinian prime minister Salam Fayyad state will set a new Government

TP: Emergency session of the Palestinian Prime Minister Salam Fayyad will set the new government

REF: state of emergency period ends ; palestinian prime minister fayyad to form new government

GT: Indian territory from south to north, one week before the start after another wet season, the provincial residents hold long drought every rain in the mood to meet the heavy rain, but did not expect rain came unexpectedly fierce, a rain disaster, roads become rivers, low-lying areas housing to make Mo in the water, transport almost paralyzed, Zhi Jin statistics about You nearly 500 people due to floods were killed.

TP: Indian territory from south to north, one week before the start have entered into the rainy season, provincial residents hold long drought to hope rain in the mood to meet the heavy rain, but did not feed rain came unexpectedly fierce, a rain disaster, roads change the river, low-lying areas housing do not water, traffic almost to a standstill, since statistics are nearly 500 people due to floods killed.

REF: the whole of india , from south to north , started to progressively enter the monsoon season a week ago . the residents of each state all greeted the heavy rains as relief at the end of a long drought , but didn't expect that the rain would come with unexpected violence , a real deluge . highways have become rivers ; houses in low-lying areas have been surbmerged in the water ; the transport system is nearly paralyzed . to date , figures show that nearly 500 people have unfortunately lost their lives to the floods .

GT: But the Taliban said in the meantime, the other a German hostages kidnapped in very poor health, began to fall into a coma and lost consciousness.

TP: But the Taliban said in the meantime, another German hostages kidnapped a very weak body fell into a coma and began to lose consciousness.

REF: but at the same time the taliban said that another german hostage who had been kidnapped was in extremely poor health , and had started to become comatose and to lose consciousness .

GT: Taliban spokesman Ahmadi told AFP in an unknown location telephone interview, said: We, through tribal elders, representatives of direct contact with South Korea.

TP: Taliban spokesman Ahmadi told AFP in an unknown location telephone interview, said: We are through tribal elders, directly with the South Korean leadership, business

REF: taliban spokesperson ahmadi said in a telephone interview by afp at an undisclosed location : we have established direct contact with the south korean delegation through tribal elders .

Figure 2: Random sample of 5 items from study in Section 4: original Google translation (GT), results of targeted paraphrasing translation process (TP), and a human reference translation.

is translated to the target and then back-translated, a comparison of the result with the original is likely to identify places where the translation process encountered difficulty.[7] Briefly, we automatically translate source F to target E, then back-translate to produce F' in the source language. We compare F and F' using TERp — which, in addition to its use as an evaluation metric, is a form of string-edit distance that identifies various categories of differences between two sentences. When at least two consecutive edits are found, we flag their smallest containing syntactic constituent as a potential source of translation difficulty.[8]

In more detail, we posit that if an area of backtranslation F' has many edits relative to original sentence F, then that area probably comes from parts of the target translation that did not represent the desired meaning in F very well. We only consider consecutive edits in certain of the TERp edit categories, specifically, deletions (D), insertions (I), and shifts (S); the two remaining categories, matches (M) and paraphrases (P), indicate that the words are identical or that the original meaning was preserved. Furthermore, we assume that while a single D, S, or I edit might be fairly meaningless, a string of at least two of those types of edits is likely to represent a substantive problem in the translation.

In order to identify reasonably meaningful paraphrase units based on potential errors, we rely on a source language constituency parser. Using the parse, we find the smallest constituent of the sentence containing all of the tokens in a particular error string. At times, these constituents can be quite large, even the entire sentence. To weed out these cases, we restrict constituent length to no more than 7 tokens.

For example, given

F **The most recent probe to visit Jupiter** was the Pluto-bound New Horizons spacecraft in late February 2007.

E La investigación más reciente fue la visita de Júpiter a Plutón de la envolvente sonda New Horizons a fines de febrero de 2007.

F' The latest research visit Jupiter was the Pluto-bound New Horizons spacecraft in late February 2007.

spans in the the bolded phrase in F would be identified, based on the TERp alignment and smallest containing constituent as shown in Figure 3.

In order to evaluate this approach, we again use NIST MT08 data, this time going in the English-to-Chinese direction since we are assuming source language resources not currently available for Chinese.[9] We used English reference 0 as the source sentence, and the original Chinese sentence as the target.[10]

The data set comprises 1,357 sentence pairs. Using the above described algorithm to automatically identify possible problem areas in the translation, with the Google Translate API providing both the translation and back-translation, we generated 1,780 potential error spans in 1,006 of the sentences, and, continuing the targeted paraphrasing process, we obtained up to three source paraphrases per span, for the problemantic spans in 1,000 of those sentences. (For six sentences, no paraphrases weres suggested for any of the problematic spans.) These yielded full-sentence paraphrase alternatives for the 1,000 sentences, which we again evaluated via an oracle study.

For this study we used the TER metric (Snover et al., 2006) rather than TERp. Comparing with the GT output, we find that TP yields a better-translated paraphrase sentence is available in 313 of the 1000 cases, or 31.3%, and for those 313 cases, TER for the oracle-best paraphrase alternative improves on the TER for the original sentence by 12.16 TER points. Also taking into account the cases where there is no improvement over the baseline, the average TER score improves by 3.8 points. The cost for human tasks in this study — just paraphrases, since identifying problematic spans was done automatically — was $117.48, or a bit under $0.12 per sentence.

[7] Exactly the same insight is behind the "source-side pseudo-referencebased feature" employed by Soricut and Echihabi (2010) in their system for predicting the trustworthiness of translations.

[8] It is possible that the difficulty so identified involves back-translation only, not translation in the original direction. If that is the case, then more paraphrasing will be done than necessary, but the quality of the TP process's output should not suffer.

[9] The Stanford parser (Klein and Manning, 2002), which we use to identify source syntactic constituents, exists for both English and Chinese, but TERp uses English resources such as WordNet in order to capture acceptable variants of expression for the same meaning. Matt Snover (personal communication) is working on extension of TERp to other languages.

[10] We chose reference 0 because on inspection these references seemed most reflective of native English grammar and usage.

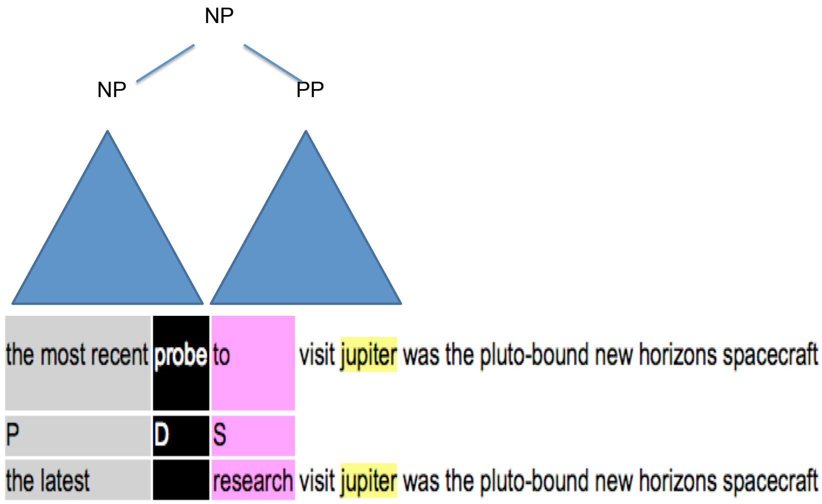

NP
NP
PP
the most recent
probe
to
visit jupiter was the pluto-bound new horizons spacecraft
P
D
S
the latest
research
visit jupiter was the pluto-bound new horizons spacecraft

Figure 3: TERp alignment of a source sentence and its back-translation in order to identify a problematic source span.

6 Conclusions and Future Work

In this paper we have focused on a relatively less-explored space on the spectrum between high quality and low cost translation: sharing the burden of the translation task among a fully automatic system and *monolingual* human participants, without requiring human bilingual expertise. The monolingual participants in this framework perform straightforward tasks: they identify parts of sentences in their language that seem to have errors, they provide sub-sentential paraphrases in context, and they judge the fluency of sentences they are presented with (or, in a variant still to be explored, they simply select which target sentence they like the best). Unlike other proposals for exploiting monolingual speakers in human-machine collaborative translation, the human steps here are amenable to automation, and in addition to evaluating a mostly-human variant of our targeted paraphrasing translation framework, we also assessed a version in which the identification of mistranslated spans (to be paraphrased) is done automatically.

Our experimentation yielded a consistent pattern of results, supporting the conclusion that targeted paraphrasing can lead to significant improvements in translation, via several different measures. First, a very small pilot study for Chinese-English translation in Wikipedia provided preliminary validation that translation fluency and accuracy can be improved quite significantly for a set of fairly chosen test sentences, according to human ratings. Second, a small experiment in Chinese-English translation using standard NIST test sentences suggested the potential for dramatic gains using the BLEU and TERp scores, with oracle improvements of 2.46 points and 5.46 points, respectively. In addition, a non-oracle experiment, selecting the best hypothesis according to the MT system's model score, yielded a gain of nearly 1.7 BLEU points. And third, in a large scale evaluation of the approach using English-Chinese translation of 1,000 sentences, this time automating the step of identifying potentially mistranslated parts of source sentences, the oracle results demonstrated that a gain of nearly 4 TER points is available.

These initial studies leave considerable room for future work. One important step will be to better characterize the relationship between cost and quality in quantitative terms: how much does it cost to obtain how much quality improvement, and how does that compare with typical professional translation costs of $0.25 per word? This question is closely connected with the dynamics of crowdsourcing platforms such as Mechanical Turk — the cost *per sentence* in these experiments works out to be around $0.12, but translation on a large scale will involve a complicated ecosystem of workers and cheaters, tasks and motivations and incentives (Quinn and Bederson, 2009). A related crowdsourcing issue requiring further study is the availability of monolingual human participants for a range of language pairs, in order to validate the argument that drawing on monolingual human participation will significantly reduce the severity of the availability bottleneck. And, of course, in the upper bound in Table 1 makes quite clear the crucial value added by bilingual translators, when they are available; we hope to explore whether the targeted paraphrasing translation pipeline can improve the productivity of post-editing by bilinguals, making it easier to move toward the upper bound in a cost-effective way.

Another set of issues concerns the underlying translation technology. A reviewer correctly notes that the value of the approach taken here is likely to vary depending upon the quality of the underlying translation system, and the approach may break down at the extrema, when the baseline translation is either already very good or completely awful. We chose to use Google Translate for its wide availability and the fact that it represents a state of the art baseline to beat; however, in future work we plan to substitute our own statistical MT systems, which will permit us to experiment across a range of translation model and language model LM training set sizes, and therefore to vary quality while keeping other system details constant. More directly connected to research in machine translation, this framework provides a variety of opportunities for improving fully automatic statistical MT systems. We plan to implement a fully automatic targeted paraphrasing translation pipeline, using the automated methods discussed when introducing the pipeline in Section 2, including translation of targeted paraphrase lattices (cf. (Max, 2010; Du et al., 2010)). Finally, we intend to explore the application of our approach in scenarios involving less-common languages, by using a more common language as a pivot or bridge (Habash and Hu, 2009).

Acknowledgments

This work has been supported in part by the National Science Foundation under awards BCS0941455 and IIS0838801. The authors would like to thank three anonymous reviewers for their helpful comments, and Chris Callison-Burch and Chris Dyer for their helpful comments and discussion.

References

Joshua S. Albrecht, Rebecca Hwa, and G. Elisabeta Marai. 2009. Correcting automatic translations through collaborations between mt and monolingual target-language users. In *EACL '09: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 60–68, Morristown, NJ, USA. Association for Computational Linguistics.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 597–604, Morristown, NJ, USA. Association for Computational Linguistics.

Benjamin B. Bederson, Chang Hu, and Philip Resnik. 2010. Translation by iterative collaboration between monolingual users. In *Graphics Interface (GI) conference*.

Lynne Bowker and Michael Barlow. 2004. Bilingual concordancers and translation memories: a comparative evaluation. In *LRTWRT '04: Proceedings of the Second International Workshop on Language Resources for Translation Work, Research and Training*, pages 70–79, Morristown, NJ, USA. Association for Computational Linguistics.

Olivia Buzek, Philip Resnik, and Ben Bederson. 2010. Error driven paraphrase annotation using mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 217–221, Los Angeles, June. Association for Computational Linguistics.

Chris Callison-Burch, Colin Bannard, , and Josh Schroeder. 2004. Improving statistical translation through editing. In *Workshop of the European Association for Machine Translation*.

Chris Callison-Burch. 2009. Fast, cheap, and creative: Evaluating translation quality using Amazon's Mechanical Turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 286–295, Singapore, August. Association for Computational Linguistics.

Jinhua Du, Jie Jiang, and Andy Way. 2010. Facilitating translation using source language paraphrase lattices.

In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, Cambridge, MA, October. Association for Computational Linguistics.

Chris Dyer and Philip Resnik. 2010. Forest translation. In *NAACL'10*.

C. Dyer, S. Muresan, and P. Resnik. 2008. Generalizing word lattice translation. In *Proceedings of HLT-ACL*, Columbus, OH.

C. Dyer. 2007. Noisier channel translation: translation from morphologically complex languages. In *Proceedings of the Second Workshop on Statistical Machine Translation*, Prague, June.

José Esteban, José Lorenzo, Antonio S. Valderrábanos, and Guy Lapalme. 2004. Transtype2 - an innovative computer-assisted translation system. In *The Companion Volume to the Proceedings of 42st Annual Meeting of the Association for Computational Linguistics*, pages 94–97, Barcelona, Spain, jul. Association for Computational Linguistics. TT2.

Nizar Habash and Jun Hu. 2009. Improving arabic-chinese statistical machine translation using english as pivot language. In *StatMT '09: Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 173–181, Morristown, NJ, USA. Association for Computational Linguistics.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Nat. Lang. Eng.*, 11(3):311–325.

Shahram Khadivi, Richard Zens, and Hermann Ney. 2006. Integration of speech to computer-assisted translation using finite-state automata. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 467–474, Morristown, NJ, USA. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2002. Fast exact inference with a factored model for natural language parsing. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15 - Neural Information Processing Systems, NIPS 2002*, pages 3–10. MIT Press.

Philipp Koehn. 2009. A web-based interactive computer aided translation tool. In *Proceedings of the ACL-IJCNLP 2009 Software Demonstrations*, pages 17–20, Suntec, Singapore, August. Association for Computational Linguistics.

Anne-Marie Laurian. 1984. Machine translation : What type of post-editing on what type of documents for what type of users. In *10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics*.

Aurélien Max. 2009. Sub-sentencial paraphrasing by contextual pivot translation. In *Proceedings of the 2009 Workshop on Applied Textual Inference*, pages 18–26, Suntec, Singapore, August. Association for Computational Linguistics.

Aurélien Max. 2010. Example-based paraphrasing for improved phrase-based statistical machine translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, Cambridge, MA, October. Association for Computational Linguistics.

Daisuke Morita and Toru Ishida. 2009. Designing protocols for collaborative translation. In *PRIMA '09: Proceedings of the 12th International Conference on Principles of Practice in Multi-Agent Systems*, pages 17–32, Berlin, Heidelberg. Springer-Verlag.

Robert Munro. 2010. Haiti emergency response: the power of crowdsourcing and SMS. Relief 2.0 in Haiti, Stanford, CA.

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alexander Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir R. Radev. 2004. A smorgasbord of features for statistical machine translation. In *HLT-NAACL*, pages 161–168.

Alex Quinn and Benjamin B. Bederson. 2009. A taxonomy of distributed human computation. Technical Report HCIL-2009-23, University of Maryland, October.

D. Shahaf and E. Horvitz. 2010. Generalized task markets for human and machine computation. In *AAAI 2010*, July.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Matt Snover, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. 2009. TER-Plus: Paraphrases, Semantic, and Alignment Enhancements to Translation Edit Rate. *Machine Translation*.

Radu Soricut and Abdessamad Echihabi. 2010. Trustrank: Inducing trust in automatic translations via ranking. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 612–621, Uppsala, Sweden, July. Association for Computational Linguistics.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *HLT '01: Proceedings of the first international conference on Human language technology research*, pages 1–8, Morristown, NJ, USA. Association for Computational Linguistics.

Soft Syntactic Constraints for Hierarchical Phrase-based Translation Using Latent Syntactic Distributions

Zhongqiang Huang
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742
zqhuang@umiacs.umd.edu

Martin Čmejrek and **Bowen Zhou**
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
{martin.cmejrek,zhou}@us.ibm.com

Abstract

In this paper, we present a novel approach to enhance hierarchical phrase-based machine translation systems with linguistically motivated syntactic features. Rather than directly using treebank categories as in previous studies, we learn a set of linguistically-guided latent syntactic categories automatically from a source-side parsed, word-aligned parallel corpus, based on the hierarchical structure among phrase pairs as well as the syntactic structure of the source side. In our model, each X nonterminal in a SCFG rule is decorated with a real-valued feature vector computed based on its distribution of latent syntactic categories. These feature vectors are utilized at decoding time to measure the similarity between the syntactic analysis of the source side and the syntax of the SCFG rules that are applied to derive translations. Our approach maintains the advantages of hierarchical phrase-based translation systems while at the same time naturally incorporates soft syntactic constraints.

1 Introduction

In recent years, syntax-based translation models (Chiang, 2007; Galley et al., 2004; Liu et al., 2006) have shown promising progress in improving translation quality, thanks to the incorporation of phrasal translation adopted from the widely used phrase-based models (Och and Ney, 2004) to handle local fluency and the engagement of synchronous context-free grammars (SCFG) to handle non-local phrase reordering. Approaches to syntax-based translation models can be largely categorized into two classes based on their dependency on annotated corpus (Chiang, 2007). Linguistically syntax-based models (e.g., (Yamada and Knight, 2001; Galley et al., 2004; Liu et al., 2006)) utilize structures defined over linguistic theory and annotations (e.g., Penn Treebank) and guide the derivation of SCFG rules with explicit parsing on at least one side of the parallel corpus. Formally syntax-based models (e.g., (Wu, 1997; Chiang, 2007)) extract synchronous grammars from parallel corpora based on the hierarchical structure of natural language pairs without any explicit linguistic knowledge or annotations. In this work, we focus on the hierarchical phrase-based models of Chiang (2007), which is formally syntax-based, and always refer the term SCFG, from now on, to the grammars of this model class.

On the one hand, hierarchical phrase-based models do not suffer from errors in syntactic constraints that are unavoidable in linguistically syntax-based models. Despite the complete lack of linguistic guidance, the performance of hierarchical phrase-based models is competitive when compared to linguistically syntax-based models. As shown in (Mi and Huang, 2008), hierarchical phrase-based models significantly outperform tree-to-string models (Liu et al., 2006; Huang et al., 2006), even when attempts are made to alleviate parsing errors using either forest-based decoding (Mi et al., 2008) or forest-based rule extraction (Mi and Huang, 2008).

On the other hand, when properly used, syntactic constraints can provide invaluable benefits to improve translation quality. The tree-to-string models of Mi and Huang (2008) can actually signif-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 138–147,
MIT, Massachusetts, USA, 9-11 October 2010.

icantly outperform hierarchical phrase-based models when using forest-based rule extraction together with forest-based decoding. Chiang (2010) also obtained significant improvement over his hierarchical baseline by using syntactic parse trees on both source and target sides to induce fuzzy (not exact) tree-to-tree rules and by also allowing syntactically mismatched substitutions.

In this paper, we augment rules in hierarchical phrase-based translation systems with novel syntactic features. Unlike previous studies (e.g., (Zollmann and Venugopal, 2006)) that directly use explicit treebank categories such as NP, NP/PP (NP missing PP from the right) to annotate phrase pairs, we induce a set of latent categories to capture the syntactic dependencies inherent in the hierarchical structure of phrase pairs, and derive a real-valued feature vector for each X nonterminal of a SCFG rule based on the distribution of the latent categories. Moreover, we convert the equality test of two sequences of syntactic categories, which are either identical or different, into the computation of a similarity score between their corresponding feature vectors. In our model, two *symbolically different* sequences of syntactic categories could have a high similarity score in the feature vector representation if they are *syntactically similar*, and a low score otherwise. In decoding, these feature vectors are utilized to measure the similarity between the syntactic analysis of the source side and the syntax of the SCFG rules that are applied to derive translations. Our approach maintains the advantages of hierarchical phrase-based translation systems while at the same time naturally incorporates soft syntactic constraints. To the best of our knowledge, this is the first work that applies real-valued syntactic feature vectors to machine translation.

The rest of the paper is organized as follows. Section 2 briefly reviews hierarchical phrase-based translation models. Section 3 presents an overview of our approach, followed by Section 4 describing the hierarchical structure of aligned phrase pairs and Section 5 describing how to induce latent syntactic categories. Experimental results are reported in Section 6, followed by discussions in Section 7. Section 8 concludes this paper.

2 Hierarchical Phrase-Based Translation

An SCFG is a synchronous rewriting system generating source and target side string pairs simultaneously based on a context-free grammar. Each synchronous production (i.e., rule) rewrites a nonterminal into a pair of strings, γ and α, where γ (or α) contains terminal and nonterminal symbols from the source (or target) language and there is a one-to-one correspondence between the nonterminal symbols on both sides. In particular, the hierarchical model (Chiang, 2007) studied in this paper explores hierarchical structures of natural language and utilize only a unified nonterminal symbol X in the grammar,

$$X \rightarrow \langle \gamma, \alpha, \sim \rangle$$

where $\sim$ is the one-to-one correspondence between X's in γ and α, and it can be indicated by underscripted co-indexes. Two example English-to-Chinese translation rules are represented as follows:

$$X \rightarrow \langle \text{give the pen to me}, \text{钢笔 给 我} \rangle \quad (1)$$

$$X \rightarrow \langle \text{give } X_1 \text{ to me}, X_1 \text{ 给 我} \rangle \quad (2)$$

The SCFG rules of hierarchical phrase-based models are extracted automatically from corpora of word-aligned parallel sentence pairs (Brown et al., 1993; Och and Ney, 2000). An aligned sentence pair is a tuple (E, F, A), where $E = e_1 \cdots e_n$ can be interpreted as an English sentence of length n, $F = f_1 \cdots f_m$ its translation of length m in a foreign language, and A a set of links between words of the two sentences. Figure 1 (a) shows an example of aligned English-to-Chinese sentence pair. Widely adopted in phrase-based models (Och and Ney, 2004), a pair of consecutive sequences of words from E and F is a *phrase pair* if all words are aligned only within the sequences and not to any word outside. We call a sequence of words a *phrase* if it corresponds to either side of a phrase pair, and a *non-phrase* otherwise. Note that the boundary words of a phrase pair may not be aligned to any other word. We call the phrase pairs with all boundary words aligned *tight* phrase pairs (Zhang et al., 2008). A tight phrase pair is the minimal phrase pair among all that share the same set of alignment links. Figure 1 (b) highlights the tight phrase pairs in the example sentence pair.

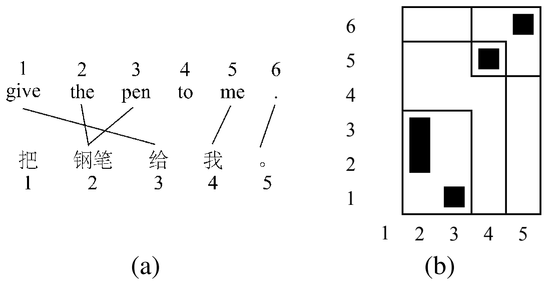


Figure 1: An example of word-aligned sentence pair (a) with tight phrase pairs marked in a matrix representation (b).

The extraction of SCFG rules proceeds as follows. In the first step, all phrase pairs below a maximum length are extracted as phrasal rules. In the second step, abstract rules are extracted from tight phrase pairs that contain other tight phrase pairs by replacing the sub phrase pairs with co-indexed X-nonterminals. Chiang (2007) also introduced several requirements (e.g., there are at most two nonterminals at the right hand side of a rule) to safeguard the quality of the abstract rules as well as keeping decoding efficient. In our example above, rule (2) can be extracted from rule (1) with the following sub phrase pair:

$$X \rightarrow \langle \text{the pen}, \text{钢笔} \rangle$$

The use of a unified X nonterminal makes hierarchical phrase-based models flexible at capturing non-local reordering of phrases. However, such flexibility also comes at the cost that it is not able to differentiate between different syntactic usages of phrases. Suppose rule $X \rightarrow \langle \text{I am reading } X_1, \cdots \rangle$ is extracted from a phrase pair with *I am reading a book* on the source side where X_1 is abstracted from the noun phrase pair . If this rule is used to translate *I am reading the brochure of a book fair*, it would be better to apply it over the entire string than over sub-strings such as *I ... the brochure of*. This is because the nonterminal X_1 in the rule was abstracted from a noun phrase on the source side of the training data and would thus be better (more informative) to be applied to phrases of the same type. Hierarchical phrase-based models are not able to distinguish syntactic differences like this.

Zollmann and Venugopal (2006) attempted to address this problem by annotating phrase pairs with treebank categories based on automatic parse trees. They introduced an extended set of categories (e.g., NP+V for *she went* and DT\NP for *great wall*, an noun phrase with a missing determiner on the left) to annotate phrase pairs that do not align with syntactic constituents. Their hard syntactic constraint requires that the nonterminals should match exactly to rewrite with a rule, which could rule out potentially correct derivations due to errors in the syntactic parses as well as to data sparsity. For example, NP cannot be instantiated with phrase pairs of type DT+NN, in spite of their syntactic similarity. Venugopal et al. (2009) addressed this problem by directly introducing soft syntactic preferences into SCFG rules using preference grammars, but they had to face the computational challenges of large preference vectors. Chiang (2010) also avoided hard constraints and took a soft alternative that directly models the cost of mismatched rule substitutions. This, however, would require a large number of parameters to be tuned on a generally small-sized held-out set, and it could thus suffer from over-tuning.

3 Approach Overview

In this work, we take a different approach to introduce linguistic syntax to hierarchical phrase-based translation systems and impose soft syntactic constraints between derivation rules and the syntactic parse of the sentence to be translated. For each phrase pair extracted from a sentence pair of a source-side parsed parallel corpus, we abstract its syntax by the sequence of highest root categories, which we call a *tag sequence*, that exactly[1] dominates the syntactic tree fragments of the source-side phrase. Figure 3 (b) shows the source-side parse tree of a sentence pair. The tag sequence for “the pen” is simply “NP” because it is a noun phrase, while phrase “give the pen” is dominated by a verb followed by a noun phrase, and thus its tag sequence is “VBP NP”.

Let $TS = \{ts_1, \cdots, ts_m\}$ be the set of all tag sequences extracted from a parallel corpus. The syntax of each X nonterminal[2] in a SCFG rule can be then

[1] In case of a non-tight phrase pair, we only abstract and compare the syntax of the largest tight part.

[2] There are three X nonterminals (one on the left and two on the right) for binary abstract rules, two for unary abstract rules, and one for phrasal rules.

Tag Sequence	Probability
NP	0.40
DT NN	0.35
DT NN NN	0.25

Table 1: The distribution of tag sequences for X_1 in $X \rightarrow \langle$I am reading $X_1, \cdots\rangle$.

characterized by the distribution of tag sequences $\vec{P}_X(TS) = (p_X(ts_1), \cdots, p_X(ts_m))$, based on the phrase pairs it is abstracted from. Table 1 shows an example distribution of tag sequences for X_1 in $X \rightarrow \langle$I am reading $X_1, \cdots\rangle$.

Instead of directly using tag sequences, as we discussed their disadvantages above, we represent each of them by a real-valued feature vector. Suppose we have a collection of n latent syntactic categories $\mathcal{C} = \{c_1, \cdots, c_n\}$. For each tag sequence ts, we compute its distribution of latent syntactic categories $\vec{P}_{ts}(\mathcal{C}) = (p_{ts}(c_1), \cdots, p_{ts}(c_n))$. For example, $\vec{P}_{\text{``NP VP''}}(\mathcal{C}) = \{0.5, 0.2, 0.3\}$ means that the latent syntactic categories c_1, c_2, and c_3 are distributed as $p(c_1) = 0.5$, $p(c_2) = 0.2$, and $p(c_3) = 0.3$ for tag sequence "NP VP". We further convert the distribution to a normalized feature vector $\vec{F}(ts)$ to represent tag sequence ts:

$$\begin{aligned}\vec{F}(ts) &= (f_1(ts), \cdots, f_n(ts)) \\ &= \frac{(p_{ts}(c_1), \cdots, p_{ts}(c_n))}{\|(p_{ts}(c_1), \cdots, p_{ts}(c_n))\|}\end{aligned}$$

The advantage of using real-valued feature vectors is that the degree of similarity between two tag sequences ts and ts' in the space of the latent syntactic categories $\mathcal{C}$ can be simply computed as a dot-product[3] of their feature vectors:

$$\vec{F}(ts) \cdot \vec{F}(ts') = \sum_{1 \leq i \leq n} f_i(ts) f_i(ts')$$

which computes a syntactic similarity score in the range of 0 (totally syntactically different) to 1 (completely syntactically identical).

Similarly, we can represent the syntax of each X nonterminal in a rule with a feature vector $\vec{F}(X)$, computed as the sum of the feature vectors of tag sequences weighted by the distribution of tag sequences of the nonterminal X:

$$\vec{F}(X) = \sum_{ts \in TS} p_X(ts) \vec{F}(ts)$$

Now we can impose soft syntactic constraints using these feature vectors when a SCFG rule is used to translate a parsed source sentence. Given that a X nonterminal in the rule is applied to a span with tag sequence[4] ts as determined by a syntactic parser, we can compute the following syntax similarity feature:

$$\text{SynSim}(X, ts) = -\log(\vec{F}(ts) \cdot \vec{F}(X))$$

Except that it is computed on the fly, this feature can be used in the same way as the regular features in hierarchical translation systems to determine the best translation, and its feature weight can be tuned in the same way together with the other features on a held-out data set.

In our approach, the set of latent syntactic categories is automatically induced from a source-side parsed, word-aligned parallel corpus based on the hierarchical structure among phrase pairs along with the syntactic parse of the source side. In what follows, we will explain the two critical aspects of our approach, i.e., how to identify the hierarchical structures among all phrase pairs in a sentence pair, and how to induce the latent syntactic categories from the hierarchy to syntactically explain the phrase pairs.

4 Alignment-based Hierarchy

The aforementioned abstract rule extraction algorithm of Chiang (2007) is based on the property that a tight phrase pair can contain other tight phrase pairs. Given two non-disjoint tight phrase pairs that share at least one common alignment link, there are only two relationships: either one completely includes another or they do not include one another but have a non-empty overlap, which we call a non-trivial overlap. In the second case, the intersection, differences, and union of the two phrase pairs are

[3] Other measures such as KL-divergence in the probability space are also feasible.

[4] A normalized uniform feature vector is used for tag sequences (of parsed test sentences) that are not seen on the training corpus.

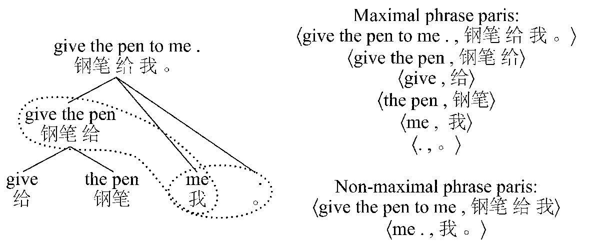


Figure 2: A decomposition tree of tight phrase pairs with all tight phrase pairs listed on the right. As highlighted, the two non-maximal phrase pairs are generated by consecutive sibling nodes.

also tight phrase pairs (see Figure 1 (b) for example), and the two phrase pairs, as well as their intersection and differences, are all sub phrase pairs of their union.

Zhang et al. (2008) exploited this property to construct a hierarchical decomposition tree (Bui-Xuan et al., 2005) of phrase pairs from a sentence pair to extract all phrase pairs in linear time. In this paper, we focus on learning the syntactic dependencies along the hierarchy of phrase pairs. Our hierarchy construction follows Heber and Stoye (2001).

Let $\mathcal{P}$ be the set of tight phrase pairs extracted from a sentence pair. We call a sequentially-ordered list[5] $L = (p_1, \cdots, p_k)$ of unique phrase pairs $p_i \in \mathcal{P}$ a *chain* if every two successive phrase pairs in L have a non-trivial overlap. A chain is *maximal* if it can not be extended to its left or right with other phrase pairs. Note that any sub-sequence of phrase pairs in a chain generates a tight phrase pair. In particular, chain L generates a tight phrase pair $\tau(L)$ that corresponds exactly to the union of the alignment links in $p \in L$. We call the phrase pairs generated by maximal chains *maximal* phrase pairs and call the other phrase pairs *non-maximal*. Non-maximal phrase pairs always overlap non-trivially with some other phrase pairs while maximal phrase pairs do not, and it can be shown that any non-maximal phrase pair can be generated by a sequence of maximal phrase pairs. Note that the largest tight phrase pair that includes all alignment links in A is also a maximal phrase pair.

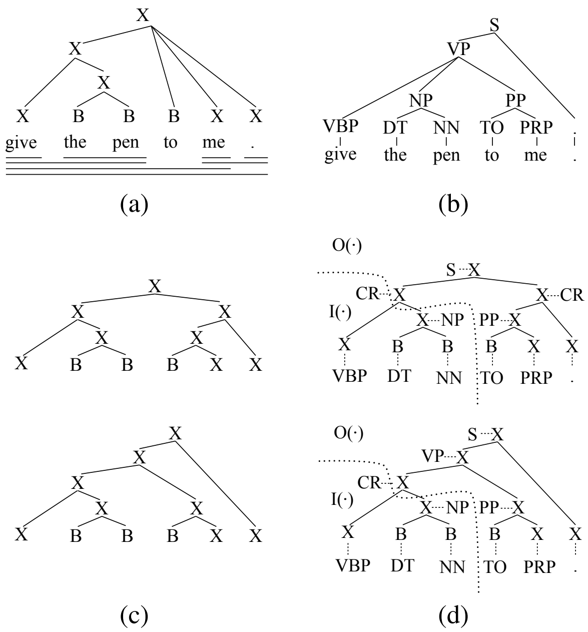


Figure 3: (a) decomposition tree for the English side of the example sentence pair with all phrases underlined, (b) automatic parse tree of the English side, (c) two example binarized decomposition trees with syntactic emissions in depicted in (d), where the two dotted curves give an example I(·) and O(·) that separate the forest into two parts.

Lemma 1 *Given two different maximal phrase pairs p_1 and p_2, exactly one of the following alternatives is true: p_1 and p_2 are disjoint, p_1 is a sub phrase pair of p_2, or p_2 is a sub phrase pair of p_1.*

A direct outcome of Lemma 1 is that there is an unique decomposition tree $T = (N, E)$ covering all of the tight phrase pairs of a sentence pair, where N is the set of maximal phrase pairs and E is the set of edges that connect between pairs of maximal phrase pairs if one is a sub phrase pair of another. All of the tight phrase pairs of a sentence pair can be extracted directly from the nodes of the decomposition tree (these phrase pairs are maximal), or generated by sequences of consecutive sibling nodes[6] (these phrase pairs are non-maximal). Figure 2 shows the decomposition tree as well as all of the tight phrase pairs that can be extracted from the example sentence pair in Figure 1.

We focus on the source side of the decomposition tree, and expand it to include all of the non-phrase

[5] The phrase pairs can be sequentially ordered first by the boundary positions of the source-side phrase and then by the boundary positions of the target-side phrase.

[6] Unaligned words may be added.

single words within the scope of the decomposition tree as frontiers and attach each as a child of the lowest node that contains the word. We then abstract the trees nodes with two symbol, X for phrases, and B for non-phrases, and call the result the *decomposition tree* of the source side phrases. Figure 3 (a) depicts such tree for the English side of our example sentence pair. We further recursively binarize[7] the decomposition tree into a binarized decomposition forest such that all phrases are directly represented as nodes in the forest. Figure 3 (c) shows two of the many binarized decomposition trees in the forest.

The binarized decomposition forest compactly encodes the hierarchical structure among phrases and non-phrases. However, the coarse abstraction of phrases with X and non-phrases with B provides little information on the constraints of the hierarchy. In order to bring in syntactic constraints, we annotate the nodes in the decomposition forest with syntactic observations based on the automatic syntactic parse tree of the source side. If a node aligns with a constituent in the parse tree, we add the syntactic category (e.g., NP) of the constituent as an emitted observation of the node, otherwise, it crosses constituent boundaries and we add a designated crossing category CR as its observation. We call the resulting forest a *syntactic decomposition forest*. Figure 3 (d) shows two syntactic decomposition trees of the forest based on the parse tree in Figure 3 (b). We will next describe how to learn finer-grained X and B categories based on the hierarchical syntactic constraints.

5 Inducing Latent Syntactic Categories

If we designate a unique symbol S as the new root of the syntactic decomposition forests introduced in the previous section, it can be shown that these forests can be generated by a probabilistic context-free grammar $G = (V, \Sigma, S, \mathcal{R}, \phi)$, where

- $V = \{S, X, B\}$ is the set of nonterminals,
- Σ is the set of terminals comprising treebank categories plus the CR tag (the crossing category),
- $S \in V$ is the unique start symbol,
- $\mathcal{R}$ is the union of the set of production rules each rewriting a nonterminal to a sequence of nonterminals and the set of emission rules each generating a terminal from a nonterminal,
- and ϕ assigns a probability score to each rule $r \in R$.

Such a grammar can be derived from the set of syntactic decomposition forests extracted from a source-side parsed parallel corpus, with rule probability scores estimated as the relative frequencies of the production and emission rules.

The X and B nonterminals in the grammar are coarse representations of phrase and non-phrases and do not carry any syntactic information at all. In order to introduce syntax to these nonterminals, we incrementally split[8] them into a set of latent categories $\{X_1, \cdots, X_n\}$ for X and another set $\{B_1, \cdots, B_n\}$ for B, and then learn a set of rule probabilities[9] ϕ on the latent categories so that the likelihood of the training forests are maximized. The motivation is to let the latent categories learn different preferences of (emitted) syntactic categories as well as structural dependencies along the hierarchy so that they can carry syntactic information. We call them *latent syntactic categories*. The learned X_i's represent syntactically-induced finer-grained categories of phrases and are used as the set of latent syntactic categories $\mathcal{C}$ described in Section 3. In related research, Matsuzaki et al. (2005) and Petrov et al. (2006) introduced latent variables to learn finer-grained distinctions of treebank categories for parsing, and Huang et al. (2009) used a similar approach to learn finer-grained part-of-speech tags for tagging. Our method is in spirit similar to these approaches.

Optimization of grammar parameters to maximize the likelihood of training forests can be achieved

[7] The intermediate binarization nodes are also labeled as either X or B based on whether they exactly cover a phrase or not.

[8] We incrementally split each nonterminal to 2, 4, 8, and finally 16 categories, with each splitting followed by several EM iterations to tune model parameters. We consider 16 an appropriate number for latent categories, not too small to differentiate between different syntactic usages and not too large for the extra computational and storage costs.

[9] Each binary production rule is now associated with a 3-dimensional matrix of probabilities, and each emission rule associated with a 1-dimensional array of probabilities.

by a variant of Expectation-Maximization (EM) algorithm. Recall that our decomposition forests are fully binarized (except the root). In the hypergraph representation (Huang and Chiang, 2005), the hyperedges of our forests all have the same format[10] $\langle (V, W), U \rangle$, meaning that node U expands to nodes V and W with production rule $U \to VW$. Given a forest F with root node R, we denote $e(U)$ the emitted syntactic category at node U and LR(U) (or PL(W), or PR(V))[11] the set of node pairs (V, W) (or (U, V), or (U, W)) such that $\langle (V, W), U \rangle$ is a hyperedge of the forest. Now consider node U, which is either S, X, or B, in the forest. Let U_x be the latent syntactic category[12] of node U. We define I(U_x) the part of the forest (includes $e(U)$ but not U_x) inside U, and O(U_x) the other part of the forest (includes U_x but not $e(U)$) outside U, as illustrated in Figure 3 (d). The inside-outside probabilities are defined as:

$$\begin{aligned} \mathrm{P}_{\mathrm{IN}}(U_x) &= P(\mathrm{I}(U_x)|U_x) \\ \mathrm{P}_{\mathrm{OUT}}(U_x) &= P(\mathrm{O}(U_x)|S) \end{aligned}$$

which can be computed recursively as:

$$\begin{aligned} \mathrm{P}_{\mathrm{IN}}(U_x) &= \sum_{(V,W)\in \mathrm{LR}(U)} \sum_{y,z} \begin{matrix} \phi(U_x \to e(U)) \\ \times \phi(U_x \to V_y W_z) \\ \times \mathrm{P}_{\mathrm{IN}}(V_y)\mathrm{P}_{\mathrm{IN}}(W_z) \end{matrix} \\ \mathrm{P}_{\mathrm{OUT}}(U_x) &= \sum_{(V,W)\in \mathrm{PL(U)}} \sum_{y,z} \begin{matrix} \phi(V_y \to e(V)) \\ \times \phi(V_y \to W_z U_x) \\ \times \mathrm{P}_{\mathrm{OUT}}(V_y)\mathrm{P}_{\mathrm{IN}}(W_z) \end{matrix} \\ &+ \sum_{(V,W)\in \mathrm{PR(U)}} \sum_{y,z} \begin{matrix} \phi(V_y \to e(V)) \\ \times \phi(V_y \to U_x W_z) \\ \times \mathrm{P}_{\mathrm{OUT}}(V_y)\mathrm{P}_{\mathrm{IN}}(W_z) \end{matrix} \end{aligned}$$

In the E-step, the posterior probability of the occurrence of production rule[13] $U_x \to V_y W_z$ is computed as:

$$P(U_x \to V_y W_z|F) = \frac{\begin{matrix} \phi(U_x \to e(U)) \\ \times \phi(U_x \to V_y W_z) \\ \times \mathrm{P}_{\mathrm{OUT}}(U_x)\mathrm{P}_{\mathrm{IN}}(V_y)\mathrm{P}_{\mathrm{IN}}(W_w) \end{matrix}}{\mathrm{P}_{\mathrm{IN}}(R)}$$

In the M-step, the expected counts of rule $U_x \to V_y W_z$ for all latent categories V_y and W_z are accumulated together and then normalized to obtain an update of the probability estimation:

$$\phi(U_x \to V_y W_z) = \frac{\#(U_x \to V_y W_z)}{\sum\limits_{(V',W')} \sum\limits_{y,z} \#(U_x \to V_y W_z)}$$

Recall that each node U labeled as X in a forest is associated with a phrase whose syntax is abstracted by a tag sequence. Once a grammar is learned, for each such node with a corresponding tag sequence ts in forest F, we compute the posterior probability that the latent category of node U being X_i as:

$$P(X_i|ts) = \frac{\mathrm{P}_{\mathrm{OUT}}(U_i)\mathrm{P}_{\mathrm{IN}}(U_i)}{\mathrm{P}_{\mathrm{IN}}(R)}$$

This contributes $P(X_i|ts)$ evidence that tag sequence ts belongs to a X_i category. When all of the evidences are computed and accumulated in $\#(X_i, ts)$, they can then be normalized to obtain the probability that the latent category of ts is X_i:

$$p_{ts}(X_i) = \frac{\#(X_i, ts)}{\sum_i \#(X_i, ts)}$$

As described in Section 3, the distributions of latent categories are used to compute the syntactic feature vectors for the SCFG rules.

6 Experiments

We conduct experiments on two tasks, English-to-German and English-to-Chinese, both aimed for speech-to-speech translation. The training data for the English-to-German task is a filtered subset of the Europarl corpus (Koehn, 2005), containing ∼300k parallel bitext with ∼4.5M tokens on each side. The dev and test sets both contain 1k sentences with one reference for each. The training data for the English-to-Chinese task is collected from transcription and human translation of conversations in travel domain. It consists of ∼500k parallel bitext with ∼3M tokens[14] on each side. Both dev and test sets contain ∼1.3k sentences, each with two references. Both

[10] The hyperedge corresponding to the root node has a different format because it is unary, but it can be handled similarly. When clear from context, we use the same variable to present both a node and its label.

[11] LR stands for the left and right children, PL for the parent and left children, and PR for the parent and right children.

[12] We never split the start symbol S, and denote $S_0 = S$.

[13] The emission rules can be handled similarly.

[14] The Chinese sentences are automatically segmented into words. However, BLEU scores are computed at character level for tuning and evaluation.

corpora are also preprocessed with punctuation removed and words down-cased to make them suitable for speech translation.

The baseline system is our implementation of the hierarchical phrase-based model of Chiang (2007), and it includes basic features such as rule and lexicalized rule translation probabilities, language model scores, rule counts, etc. We use 4-gram language models in both tasks, and conduct minimum-error-rate training (Och, 2003) to optimize feature weights on the dev set. Our baseline hierarchical model has 8.3M and 9.7M rules for the English-to-German and English-to-Chinese tasks, respectively.

The English side of the parallel data is parsed by our implementation of the Berkeley parser (Huang and Harper, 2009) trained on the combination of Broadcast News treebank from Ontonotes (Weischedel et al., 2008) and a speechified version of the WSJ treebank (Marcus et al., 1999) to achieve higher parsing accuracy (Huang et al., 2010). Our approach introduces a new syntactic feature and its feature weight is tuned in the same way together with the features in the baseline model. In this study, we induce 16 latent categories for both X and B nonterminals.

Our approach identifies $\sim$180k unique tag sequences for the English side of phrase pairs in both tasks. As shown by the examples in Table 2, the syntactic feature vector representation is able to identify similar and dissimilar tag sequences. For instance, it determines that the sequence of "DT JJ NN" is syntactically very similar to "DT ADJP NN" while very dissimilar to "NN CD VP". Notice that our latent categories are learned automatically to maximize the likelihood of the training forests extracted based on alignment and are not explicitly instructed to discriminate between syntactically different tag sequences. Our approach is not guaranteed to always assign similar feature vectors to syntactically similar tag sequences. However, as the experimental results show below, the latent categories are able to capture some similarities among tag sequences that are beneficial for translation.

Table 3 and 4 report the experimental results on the English-to-German and English-to-Chinese tasks, respectively. The addition of the syntax feature achieves a statistically significant improvement ($p \leq 0.01$) of 0.6 in BLEU on the test set of the English-to-German task. This improvement is substantial given that only one reference is used for each test sentence. On the English-to-Chinese task, the syntax feature achieves a smaller improvement of 0.41 BLEU on the test set. One potential explanation for the smaller improvement is that the sentences on the English-to-Chinese task are much shorter, with an average of only 6 words per sentence, compared to 15 words in the English-to-German task. The hypothesis space of translating a longer sentence is much larger than that of a shorter sentence. Therefore, there is more potential gain from using syntax features to rule out unlikely derivations of longer sentences, while phrasal rules might be adequate for shorter sentences, leaving less room for syntax to help as in the case of the English-to-Chinese task.

	Baseline	**+Syntax**	**Δ**
Dev	16.26	17.06	0.80
Test	16.41	17.01	0.60

Table 3: BLEU scores of the English-to-German task (one reference).

	Baseline	**+Syntax**	**Δ**
Dev	46.47	47.39	0.92
Test	45.45	45.86	0.41

Table 4: BLEU scores of the English-to-Chinese task (two references).

7 Discussions

The incorporation of the syntactic feature into the hierarchical phrase-based translation system also brings in additional memory load and computational cost. In the worst case, our approach requires storing one feature vector for each tag sequence and one feature vector for each nonterminal of a SCFG rule, with the latter taking the majority of the extra memory storage. We observed that about 90% of the X nonterminals in the rules only have one tag sequence, and thus the required memory space can be significantly reduced by only storing a pointer to the feature vector of the tag sequence for these nonterminals. Our approach also requires computing one dot-product of two feature vectors for each nonterminal when a SCFG rule is applied to a source span.

	Very similar	**Not so similar**	**Very dissimilar**
	$\vec{F}(ts) \cdot \vec{F}(ts') > 0.9$	$0.4 \leq \vec{F}(ts) \cdot \vec{F}(ts') \leq 0.6$	$\vec{F}(ts) \cdot \vec{F}(ts') < 0.1$
DT JJ NN	DT NN	DT JJ JJ NML NN	PP NP NN
	DT JJ JJ NN	DT JJ CC INTJ VB	NN CD VP
	DT ADJP NN	DT NN NN JJ	RB NP IN CD
VP	VB	VP PP JJ NN	JJ NN TO VP
	VB RB VB PP	VB NN NN VB	JJ WHNP DT NN
	VB DT DT NN	VB RB IN JJ	IN INTJ NP
ADJP	JJ	ADJP JJ JJ CC	ADJP IN NP JJ
	PDT JJ	ADJP VB JJ JJ	AUX RB ADJP
	RB JJ	ADVP WHNP JJ	ADJP VP

Table 2: Examples of similar and dissimilar tag sequences.

This cost can be reduced, however, by caching the dot-products of the tag sequences that are frequently accessed.

There are other successful investigations to impose soft syntactic constraints to hierarchical phrase-based models by either introducing syntax-based rule features such as the prior derivation model of Zhou et al. (2008) or by imposing constraints on translation spans at decoding time, e.g., (Marton and Resnik, 2008; Xiong et al., 2009; Xiong et al., 2010). These approaches are all orthogonal to ours and it is expected that they can be combined with our approach to achieve greater improvement.

This work is an initial effort to investigate latent syntactic categories to enhance hierarchical phrase-based translation models, and there are many directions to continue this line of research. First, while the current approach imposes soft syntactic constraints between the parse structure of the source sentence and the SCFG rules used to derive the translation, the real-valued syntactic feature vectors can also be used to impose soft constraints between SCFG rules when rule rewrite occurs. In this case, target side parse trees could also be used alone or together with the source side parse trees to induce the latent syntactic categories. Second, instead of using single parse trees during both training and decoding, our approach is likely to benefit from exploring parse forests as in (Mi and Huang, 2008). Third, in addition to the treebank categories obtained by syntactic parsing, lexical cues directly available in sentence pairs could also to explored to guide the learning of latent categories. Last but not the least, it would be interesting to investigate discriminative training approaches to learn latent categories that directly optimize on translation quality.

8 Conclusion

We have presented a novel approach to enhance hierarchical phrase-based machine translation systems with real-valued linguistically motivated feature vectors. Our approach maintains the advantages of hierarchical phrase-based translation systems while at the same time naturally incorporates soft syntactic constraints. Experimental results showed that this approach improves the baseline hierarchical phrase-based translation models on both English-to-German and English-to-Chinese tasks. We will continue this line of research and exploit better ways to learn syntax and apply syntactic constraints to machine translation.

Acknowledgements

This work was done when the first author was visiting IBM T. J. Watson Research Center as a research intern. We would like to thank Mary Harper for lots of insightful discussions and suggestions and the anonymous reviewers for the helpful comments.

References

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*.

Binh Minh Bui-Xuan, Michel Habib, and Christophe Paul. 2005. Revisiting T. Uno and M. Yagiura's algorithm. In *ISAAC*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*.

David Chiang. 2010. Learning to translate with source and target syntax. In *ACL*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2004. What's in a translation rule. In *HLT/NAACL*.

Steffen Heber and Jens Stoye. 2001. Finding all common intervals of k permutations. In *CPM*.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *International Workshop on Parsing Technology*.

Zhongqiang Huang and Mary Harper. 2009. Self-training PCFG grammars with latent annotations across languages. In *EMNLP*.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. A syntax-directed translator with extended domain of locality. In *CHSLP*.

Zhongqiang Huang, Vladimir Eidelman, and Mary Harper. 2009. Improving a simple bigram hmm part-of-speech tagger by latent annotation and self-training. In *NAACL*.

Zhongqiang Huang, Mary Harper, and Slav Petrov. 2010. Self-training with products of latent variable. In *EMNLP*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *ACL*.

Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor, 1999. *Treebank-3*. Linguistic Data Consortium, Philadelphia.

Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *ACL*.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *ACL*.

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *EMNLP*.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *ACL*.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *ACL*.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL*.

Ashish Venugopal, Andreas Zollmann, Noah A. Smith, and Stephan Vogel. 2009. Preference grammars: softening syntactic constraints to improve statistical machine translation. In *NAACL*.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, and Ann Houston, 2008. *OntoNotes Release 2.0*. Linguistic Data Consortium, Philadelphia.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*.

Deyi Xiong, Min Zhang, Aiti Aw, and Haizhou Li. 2009. A syntax-driven bracketing model for phrase-based translation. In *ACL-IJCNLP*.

Deyi Xiong, Min Zhang, and Haizhou Li. 2010. Learning translation boundaries for phrase-based decoding. In *NAACL-HLT*.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *ACL*.

Hao Zhang, Daniel Gildea, and David Chiang. 2008. Extracting synchronous grammar rules from word-level alignments in linear time. In *COLING*.

Bowen Zhou, Bing Xiang, Xiaodan Zhu, and Yuqing Gao. 2008. Prior derivation models for formally syntax-based translation using linguistically syntactic parsing and tree kernels. In *SSST*.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *StatMT*.

A Hybrid Morpheme-Word Representation for Machine Translation of Morphologically Rich Languages[*]

Minh-Thang Luong **Preslav Nakov** **Min-Yen Kan**
Department of Computer Science
National University of Singapore
13 Computing Drive
Singapore 117417
{luongmin,nakov,kanmy}@comp.nus.edu.sg

Abstract

We propose a language-independent approach for improving statistical machine translation for morphologically rich languages using a hybrid morpheme-word representation where the basic unit of translation is the morpheme, but word boundaries are respected at all stages of the translation process. Our model extends the classic phrase-based model by means of (1) word boundary-aware morpheme-level phrase extraction, (2) minimum error-rate training for a morpheme-level translation model using word-level BLEU, and (3) joint scoring with morpheme- and word-level language models. Further improvements are achieved by combining our model with the classic one. The evaluation on English to Finnish using *Europarl* (714K sentence pairs; 15.5M English words) shows statistically significant improvements over the classic model based on BLEU and human judgments.

1 Introduction

The fast progress of statistical machine translation (SMT) has boosted translation quality significantly. While research keeps diversifying, *the word* remains the atomic token-unit of translation. This is fine for languages with limited morphology like English and French, or no morphology at all like Chinese, but it is inadequate for morphologically rich languages like Arabic, Czech or Finnish (Lee, 2004; Goldwater and McClosky, 2005; Yang and Kirchhoff, 2006).

There has been a line of recent SMT research that incorporates morphological analysis as part of the translation process, thus providing access to the information within the individual words. Unfortunately, most of this work either relies on language-specific tools, or only works for very small datasets.

Below we propose a language-independent approach to SMT of morphologically rich languages using a hybrid morpheme-word representation where the basic unit of translation is the morpheme, but word boundaries are respected at all stages of the translation process. We use unsupervised morphological analysis and we incorporate its output into the process of translation, as opposed to relying on pre-processing and post-processing only as has been done in previous work.

The remainder of the paper is organized as follows. Section 2 reviews related work. Sections 3 and 4 present our morphological and phrase merging enhancements. Section 5 describes our experiments, and Section 6 analyzes the results. Finally, Section 7 concludes and suggests directions for future work.

2 Related Work

Most previous work on morphology-aware approaches relies heavily on language-specific tools, e.g., the *TreeTagger* (Schmid, 1994) or the *Buckwalter* Arabic Morphological Analyzer (Buckwalter, 2004), which hampers their portability to other languages. Moreover, the prevalent method for incorporating morphological information is by heuristically-driven pre- or post-processing. For example, Sadat and Habash (2006) use different combinations of Arabic pre-processing schemes

[*] This research was sponsored in part by CSIDM (grant # 200805) and by a National Research Foundation grant entitled "Interactive Media Search" (grant # R-252-000-325-279).

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 148–157,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

for Arabic-English SMT, whereas Oflazer and El-Kahlout (2007) post-processes Turkish morpheme-level translations by re-scoring n-best lists with a word-based language model. These systems, however, do not attempt to incorporate their analysis as part of the decoding process, but rather rely on models designed for word-token translation.

We should also note the importance of the translation direction: it is much harder to translate from a morphologically poor to a morphologically rich language, where morphological distinctions not present in the source need to be generated in the target language. Research in translating into morphologically rich languages, has attracted interest for languages like *Arabic* (Badr et al., 2008), *Greek* (Avramidis and Koehn, 2008), *Hungarian* (Novák, 2009; Koehn and Haddow, 2009), *Russian* (Toutanova et al., 2008), and *Turkish* (Oflazer and El-Kahlout, 2007). These approaches, however, either only succeed in enhancing the performance for small bi-texts (Badr et al., 2008; Oflazer and El-Kahlout, 2007), or improve only modestly for large bi-texts[1].

3 Morphological Enhancements

We present a morphologically-enhanced version of the classic phrase-based SMT model (Koehn et al., 2003). We use a hybrid morpheme-word representation where the basic unit of translation is the morpheme, but word boundaries are respected at all stages of the translation process. This is in contrast with previous work, where morphological enhancements are typically performed as pre-/post-processing steps only.

In addition to changing the basic translation token unit from a word to a morpheme, our model extends the phrase-based SMT model with the following:

1. word boundary-aware morpheme-level phrase extraction;
2. minimum error-rate training for a morpheme-level model using word-level BLEU;
3. joint scoring with morpheme- and word-level language models.

We first introduce our morpheme-level representation, and then describe our enhancements.

3.1 Morphological Representation

Our morphological representation is based on the output of an unsupervised morphological analyzer. Following Virpioja et al. (2007), we use *Morfessor*, which is trained on raw tokenized text (Creutz and Lagus, 2007). The tool segments words into morphemes annotated with the following labels: PRE (prefix), STM (stem), SUF (suffix). Multiple prefixes and suffixes can be proposed for each word; word compounding is allowed as well. The output can be described by the following regular expression:

$$\texttt{WORD} = (\ \texttt{PRE*}\ \ \texttt{STM}\ \ \texttt{SUF*}\ )^{+}$$

For example, uncarefully is analyzed as

un/PRE+ care/STM+ ful/SUF+ ly/SUF

The above token sequence forms the input to our system. We keep the PRE/STM/SUF tags as part of the tokens, and distinguish between care/STM+ and care/STM. Note also that the “+” sign is appended to each nonfinal tag so that we can distinguish word-internal from word-final morphemes.

3.2 Word Boundary-aware Phrase Extraction

The core translation structure of a phrase-based SMT model is the *phrase table*, which is learned from a bilingual parallel sentence-aligned corpus, typically using the alignment template approach (Och and Ney, 2004). It contains a set of bilingual phrase pairs, each associated with five scores: forward and backward phrase translation probabilities, forward and backward lexicalized translation probabilities, and a constant phrase penalty.

The maximum phrase length n is normally limited to seven words; higher values of n increase the table size exponentially without actually yielding performance benefit (Koehn et al., 2003). However, things are different when translating with morphemes, for two reasons: (1) morpheme-token phrases of length n can span less than n words; and (2) morpheme-token phrases may only partially span words.

The first point means that morpheme-token phrase pairs span fewer word tokens, and thus cover a smaller context, which may result in fewer total extracted pairs compared to a word-level approach. Figure 1 shows a case where three Finnish words consist of nine morphemes. Previously, this issue was addressed by simply increasing the value of n when using morphemes, which is of limited help.

[1] Avramidis and Koehn (2008) improved by 0.15 BLEU over a 18.05 English-Greek baseline; Toutanova et al. (2008) improved by 0.72 BLEU over a 36.00 English-Russian baseline.

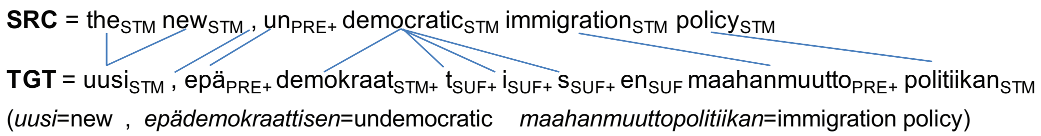

SRC = the$_{STM}$ new$_{STM}$, un$_{PRE+}$ democratic$_{STM}$ immigration$_{STM}$ policy$_{STM}$
TGT = uusi$_{STM}$, epä$_{PRE+}$ demokraat$_{STM+}$ t$_{SUF+}$ i$_{SUF+}$ s$_{SUF+}$ en$_{SUF}$ maahanmuutto$_{PRE+}$ politiikan$_{STM}$
(uusi=new , epädemokraattisen=undemocratic maahanmuuttopolitiikan=immigration policy)

Figure 1: **Example of English-Finnish bilingual fragments morphologically segmented by *Morfessor*.** Solid links represent IBM Model 4 alignments at the morpheme-token level. Translation glosses for Finnish are given below.

The second point is more interesting: morpheme-level phrases may span words partially, making them potentially usable in translating unknown inflected forms of known source language words, but also creates the danger of generating sequences of morphemes that are not legal target language words.

For example, let us consider the phrase in Figure 1: $\mathtt{un}_{\mathtt{PRE+}}$ $\mathtt{democratic}_{\mathtt{STM}}$. The original algorithm will extract the spurious phrase $\mathtt{epä}_{\mathtt{PRE+}}$ $\mathtt{demokraat}_{\mathtt{STM+}}$ $\mathtt{t}_{\mathtt{SUF+}}$ $\mathtt{i}_{\mathtt{SUF+}}$ $\mathtt{s}_{\mathtt{SUF+}}$, beside the correct one that has $\mathtt{en}_{\mathtt{SUF}}$ appended at the end. Such a spurious phrase does not generally help in translating unknown inflected forms, especially for morphologically-rich languages that feature multiple affixes, but negatively affects the translation model in terms of complexity and quality.

We solve both problems by modifying the phrase-pair extraction algorithm so that morpheme-token phrases can extend longer than n, as long as they span n words or less. We further require that word boundaries be respected[2], i.e., morpheme-token phrases span a sequence of whole words. This is a fair extension of the morpheme-token system with respect to a word-token one since both are restricted to span up to n word-tokens.

3.3 Morpheme-Token MERT Optimizing Word-Token BLEU

Modern phrase-based SMT systems use a log-linear model with the following typical feature functions: language model probabilities, word penalty, distortion cost, and the five parameters from the phrase table. Their weights are set by optimizing BLEU score (Papineni et al., 2001) directly using minimum error rate training (MERT), as suggested by Och (2003).

In previous work, phrase-based SMT systems using morpheme-token input/output naturally performed MERT at the morpheme-token level as well. This is not optimal since the final expected system output is a sequence of words, not morphemes. The main danger is that optimizing a morpheme-token BLEU score could lead to a suboptimal weight for the word penalty feature function: this is because the brevity penalty of BLEU is calculated with respect to the number of morphemes, which may vary for sentences with an identical number of words.

This motivates us to perform MERT at the word-token level, although our input consists of morphemes. In particular, for each iteration of MERT, as soon as the decoder generates a morpheme-token translation for a sentence, we convert it into a word-token sequence, which is used to calculate BLEU. We thus achieve MERT optimization at the word-token level while translating a morpheme-token input and generating a morpheme-token output.

3.4 Scoring with Twin Language Models

An SMT system that takes morpheme-token input and generates morpheme-token output should naturally use a morpheme-token language model (LM). This has the advantage of alleviating the problem of data sparseness, especially when translating into a morphologically rich language, since the LM would be able to handle some new unseen inflected forms of known words. On the negative side, a morpheme-token LM spans fewer word-tokens and thus has a more limited word "horizon" compared to one operating at the word level. As with the maximum phrase length, mechanically increasing the order of the morpheme-token LM has a limited impact.

In order to address the issue in a more principled manner, we enhance our model with a second LM that works at the word-token level. This LM is used together with the morpheme-token LM, which is achieved by using two separate feature functions in the log-linear SMT model: one for each LM. We further had to modify the Moses decoder so that

[2] This means that we miss the opportunity to generate new wordforms for known baseforms, but removes the problem of proposing nonwords in the target language.

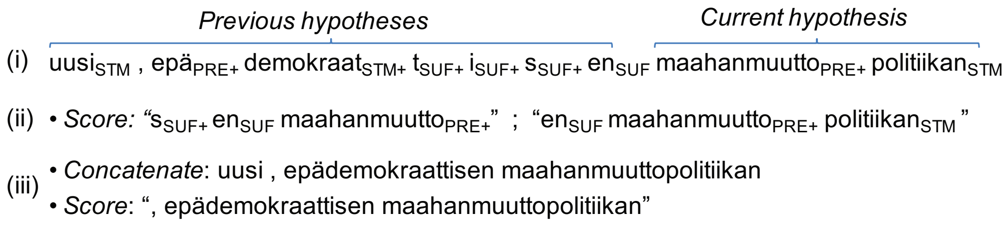

Previous hypotheses
Current hypothesis
(i) uusi_{STM} , epä_{PRE+} demokraat_{STM+} t_{SUF+} i_{SUF+} s_{SUF+} en_{SUF} $\text{maahanmuutto}_{PRE+}$ politiikan_{STM}
(ii) • Score: "s_{SUF+} en_{SUF} $\text{maahanmuutto}_{PRE+}$" ; "$\text{en}_{SUF}$ $\text{maahanmuutto}_{PRE+}$ politiikan_{STM} "
(iii) • Concatenate: uusi , epädemokraattisen maahanmuuttopolitiikan
• Score: ", epädemokraattisen maahanmuuttopolitiikan"

Figure 2: **Scoring with twin LMs.** Shown are: (i) The current state of the decoding process with the target phrases covered by the current partial hypotheses. (ii, iii) Scoring with 3-gram morpheme-token and 3-gram word-token LMs, respectively. For the word-token LM, the morpheme-token sequence is concatenated into word-tokens before scoring.

it can be enhanced with an appropriate word-token "view" on the partial morpheme-level hypotheses[3].

The interaction of the twin LMs is illustrated in Figure 2. The word-token LM can capture much longer phrases and more complete contexts such as ", *epädemokraattisen maahanmuuttopolitiikan*" compared to the morpheme-token LM.

Note that scoring with two LMs that see the output sequence as different numbers of tokens is not readily offered by the existing SMT decoders. For example, the phrase-based model in Moses (Koehn et al., 2007) allows scoring with multiple LMs, but assumes they use the same token granularity, which is useful for LMs trained on different monolingual corpora, but cannot handle our case. While the factored translation model (Koehn and Hoang, 2007) in Moses does allow scoring with models of different granularity, e.g., lemma-token and word-token LMs, it requires a 1:1 correspondence between the tokens in the different factors, which clearly is not our case.

Note that scoring with twin LMs is conceptually superior to n-best re-scoring with a word-token LM, e.g., (Oflazer and El-Kahlout, 2007), since it is tightly integrated into decoding: it scores partial hypotheses and influenced the search process directly.

4 Enriching the Translation Model

Another general strategy for combining evidence from the word-token and the morpheme-token representations is to build two separate SMT systems and then combine them. This can be done as a post-processing system combination step; see (Chen et al., 2009a) for an overview of such approaches. However, for phrase-based SMT systems, it is theoretically more appealing to combine their phrase tables since this allows the translation models of both systems to influence the hypothesis search directly.

We now describe our phrase table combination approach. Note that it is orthogonal to the work presented in the previous section, which suggests combining the two (which we will do in Section 5).

4.1 Building a Twin Translation Model

Figure 3 shows a general scheme of our twin translation model. First, we tokenize the input at different granularities: (1) morpheme-token and (2) word-token. We then build separate phrase tables (PT) for the two inputs: a word-token PT_w and a morpheme-token PT_m. Second, we re-tokenize PT_w at the morpheme level, thus obtaining a new phrase table $PT_{w\rightarrow m}$, which is of the same granularity as PT_m. Finally, we merge $PT_{w\rightarrow m}$ and PT_m, and we input the resulting phrase table to the decoder.

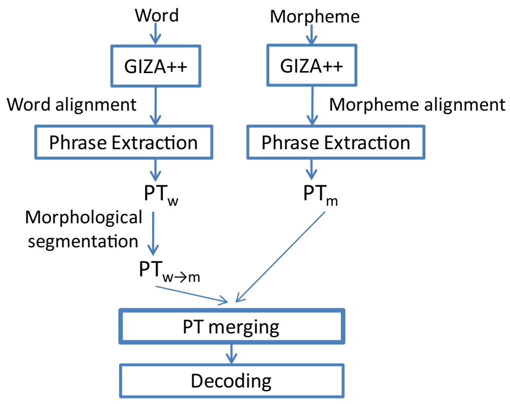


Figure 3: **Building a twin phrase table (PT).** First, separate PTs are generated for different input granularities: word-token and morpheme-token. Second, the word-token PT is retokenized at the morpheme-token level. Finally, the two PTs are merged and used by the decoder.

[3] We use the term "hypothesis" to collectively refer to the following (Koehn, 2003): the *source phrase* covered, the corresponding *target phrase*, and most importantly, a *reference to the previous hypothesis* that it extends.

4.2 Merging and Normalizing Phrase Tables

Below we first describe the two general phrase table combination strategies used in previous work: (1) direct merging using additional feature functions, and (2) phrase table interpolation. We then introduce our approach.

Add-feature methods. The first line of research on phrase table merging is exemplified by (Niehues et al., 2009; Chen et al., 2009b; Do et al., 2009; Nakov and Ng, 2009). The idea is to select one of the phrase tables as primary and to add to it all non-duplicating phrase pairs from the second table together with their associated scores. For each entry, features can be added to indicate its origin (whether from the primary or from the secondary table). Later in our experiments, we will refer to these baseline methods as *add-1* and *add-2*, depending on how many additional features have been added. The values we used for these features in the baseline are given in Section 5.4; their weights in the log-linear model were set in the standard way using MERT.

Interpolation-based methods. A problem with the above method is that the scores in the merged phrase table that correspond to forward and backward phrase translation probabilities, and forward and backward lexicalized translation probabilities can no longer be interpreted as probabilities since they are not normalized any more. Theoretically, this is not necessarily a problem since the log-linear model used by the decoder does not assume that the scores for the feature functions come from a normalized probability distribution. While it is possible to re-normalize the scores to convert them into probabilities, this is rarely done; it also does not solve the problem with the dropped scores for the duplicated phrases. Instead, the conditional probabilities in the two phrase tables are often interpolated directly, e.g., using linear interpolation. Representative work adopting this approach is (Wu and Wang, 2007). We refer to this method as *interpolation*.

Our method. The above phrase merging approaches have been proposed for phrase tables derived from different sources. This is in contrast with our twin translation scenario, where the morpheme-token phrase tables are built from the same training dataset; the main difference being that word alignments and phrase extraction were performed at the word-token level for $PT_{w\rightarrow m}$ and at the morpheme-token level for PT_m. Thus, we propose different merging approaches for the phrase translation probabilities and for the lexicalized probabilities.

In phrase-based SMT, phrase translation probabilities are computed using maximum likelihood (ML) estimation $\phi(\bar{f}|\bar{e}) = \frac{\#(\bar{f},\bar{e})}{\sum_{\bar{f}} \#(\bar{f},\bar{e})}$, where $\#(\bar{f},\bar{e})$ is the number of times the pair $(\bar{f},\bar{e})$ is extracted from the training dataset (Koehn et al., 2003). In order to preserve the normalized ML estimations as much as possible, we refrain from interpolation. Instead, we use the raw counts for the two models $\#_m(\bar{f},\bar{e})$ and $\#_{w\rightarrow m}(\bar{f},\bar{e})$ directly as follows:

$$\phi(\bar{f},\bar{e}) = \frac{\#_m(\bar{f},\bar{e}) + \#_{w\rightarrow m}(\bar{f},\bar{e})}{\sum_{\bar{f}} \#_m(\bar{f},\bar{e}) + \sum_{\bar{f}} \#_{w\rightarrow m}(\bar{f},\bar{e})}$$

For lexicalized translation probabilities, we would like to use simple interpolation. However, we notice that when a phrase pair belongs to only one of the phrase tables, the corresponding lexicalized score for the other table would be zero. This might cause some good phrases to be penalized just because they were not extracted in both tables, which we want to prevent. We thus perform interpolation from PT_m and PT_w according to the following formula:

$$\begin{aligned} \text{lex}(\bar{f}|\bar{e}) &= \alpha \times \text{lex}_m(\bar{f}_m|\bar{e}_m) \\ &+ (1-\alpha) \times \text{lex}_w(\bar{f}_w|\bar{e}_w) \end{aligned}$$

where the concatenation of $\bar{f}_m$ and $\bar{e}_m$ into word-token sequences yields $\bar{f}_w$ and $\bar{e}_w$, respectively.

If both $(\bar{f}_m, \bar{e}_m)$ and $(\bar{f}_w, \bar{e}_w)$ are present in PT_m and PT_w, respectively, we have a simple interpolation of their corresponding lexicalized scores lex$_m$ and lex$_w$. However, if one of them is missing, we do not use a zero for its corresponding lexicalized score, but use an estimate as follows.

For example, if only the entry $(\bar{f}_m, \bar{e}_m)$ is present in PT_m, we first convert ($\bar{f}_m$,$\bar{e}_m$) into a word-token pair ($\bar{f}_{m\rightarrow w}$,$\bar{e}_{m\rightarrow w}$), and then induce a corresponding word alignment from the morpheme-token alignment of ($\bar{f}_m$,$\bar{e}_m$). We then estimate a lexicalized phrase score using the original formula given in (Koehn et al., 2003), where we plug this induced word alignment and word-token lexical translation probabilities estimated from the word-token dataset The case when $(\bar{f}_w, \bar{e}_w)$ is present in PT_w, but $(\bar{f}_m, \bar{e}_m)$ is not, is solved similarly.

5 Experiments and Evaluation

5.1 Datasets

In our experiments, we use the English-Finnish data from the 2005 shared task (Koehn and Monz, 2005), which is split into training, development, and test portions; see Table 1 for details. We further split the training dataset into four subsets T_1, T_2, T_3, and T_4 of sizes 40K, 80K, 160K, and 320K parallel sentence pairs, which we use for studying the impact of training data size on translation performance.

	Sent.	Avg. words		Avg. morph.	
		en	fi	en	fi
Train	714K	21.62	15.80	24.68	26.15
Dev	2K	29.33	20.99	33.40	34.94
Test	2K	28.98	20.72	33.10	34.47

Table 1: **Dataset statistics.** Shown are the number of parallel sentences, and the average number of words and *Morfessor* morphemes on the English and Finnish sides of the training, development and test datasets.

5.2 Baseline Systems

We build two phrase-based baseline SMT systems, both using Moses (Koehn et al., 2007):

w-system: works at the word-token level, extracts phrases of up to seven words, and uses a 4-gram word-token LM (as typical for phrase-based SMT);

m-system: works at the morpheme level, tokenized using *Morfessor*[4] and augmented with "+" as described in Section 3.1.

Following Oflazer and El-Kahlout (2007) and Virpioja et al. (2007), we use phrases of up to 10 morpheme-tokens and a 5-gram morpheme-token LM. None of the enhancements described previously is applied yet. After decoding, morphemes are concatenated back to words using the "+" markers.

To evaluate the translation quality, we compute BLEU (Papineni et al., 2001) at the word-token level. We further introduce a morpheme-token version of BLEU, which we call m-BLEU: it first segments the system output and the reference translation into morpheme-tokens and then calculates a BLEU score as usual. Table 2 shows the baseline results. We can see that the *m-system* achieves much higher m-BLEU scores, indicating that it may have better morpheme coverage[5]. However, the *m-system* is outperformed by the *w-system* on the classic word-token BLEU, which means that it either does not perform as well as the *w-system* or that word-token BLEU is not capable of measuring the morpheme-level improvements. We return to this question later.

	w-system		m-system	
	BLEU	m-BLEU	BLEU	m-BLEU
T_1	11.56	45.57	11.07	49.15
T_2	12.95	48.63	12.68	53.78
T_3	13.64	50.30	13.32	54.40
T_4	14.20	50.85	13.57	54.70
Full	14.58	53.05	14.08	55.26

Table 2: **Baseline system performance** (on the test dataset). Shown are word BLEU and morpheme m-BLEU scores for the *w-system* and *m-system*.

5.3 Adding Morphological Enhancements

We now add our three morphological enhancements from Section 3 to the baseline *m-system*:

phr (training) allow morpheme-token phrases to get potentially longer than seven morpheme-tokens as long as they cover no more than seven words;

tune (tuning) MERT for morpheme-token translations while optimizing word-token BLEU;

lm (decoding) scoring morpheme-token translation hypotheses with a 5-gram morpheme-token and a 4-gram word-token LM.

The results are shown in Table 3 (ii). As we can see, each of the three enhancements yields improvements in BLEU score over the *m-system*, both for small and for large training corpora. In terms of performance ranking, *tune* achieves the best absolute improvement of 0.66 BLEU points on T_1 and of 0.47 points on the full dataset, followed by *lm* and *phr*.

Table 3 (iii) further shows that using *phr* and *lm* together yields absolute improvements of 0.70 BLEU points on T_1 and 0.50 points on the full training dataset. Further incorporating *tune*, however, only helps when training on T_1.

Overall, the morphological enhancements are on par with the *w-system* baseline, and yield sizable im-

[4] We retrained *Morfessor* for Finnish/English on the Finnish/English side of the training dataset.

[5] Note that these morphemes were generated automatically and thus many of them are erroneous.

	System	T_1 (40K)	Full (714K)
(i)	*w-system* (*w*)	11.56	14.58
	m-system (*m*)	11.07	14.08
(ii)	*m+phr*	$11.44^{+0.37}$	$14.43^{+0.35}$
	m+tune	$11.73^{+0.66}$	$14.55^{+0.47}$
	m+lm	$11.58^{+0.51}$	$14.53^{+0.45}$
(iii)	*m+phr+lm*	$11.77^{+0.70}$	**14.58**$^{+0.50}$
	m+phr+lm+tune	**11.90**$^{+0.83}$	$14.39^{+0.31}$

Table 3: **Impact of the morphological enhancements** (on test dataset). Shown are BLEU scores (in %) for training on T_1 and on the full dataset for (i) baselines, (ii) enhancements individually, and (iii) combined. Superscripts indicate absolute improvements w.r.t *m-system*.

provements over the *m-system* baseline: 0.83 BLEU points on T_1 and 0.50 on the full training dataset.

5.4 Combining Translation Tables

Finally, we investigate the effect of combining phrase tables derived from a word-token and a morpheme-token input, as described in Section 4. We experiment with the following merging methods:

add-1: phrase table merging using one table as primary and adding *one* extra feature[6];

add-2: phrase table merging using one table as primary and adding *two* extra features[7];

interpolation: simple linear interpolation with one parameter α;

ourMethod: our interpolation-like merging method described in Section 4.2.

Parameter tuning. We tune the parameters of the above methods on the development dataset.

	T_1 (40K)	Full (714K)
PT_m is primary	11.99	13.45
$PT_{w\to m}$ is primary	12.26	14.19

Table 4: **Effect of selection of primary phrase table for *add-1*** (on dev dataset): $PT_{w\to m}$, derived from a word-token input, vs. PT_m, from a morpheme-token input. Shown is BLEU (in %) on T_1 and the full training dataset.

For *add-1* and *add-2*, we need to decide which ($PT_{w\to m}$ or PT_m) phrase table should be considered the primary table. Table 4 shows the results when trying both strategies on *add-1*. As we can see, using $PT_{w\to m}$ as primary performs better on T_1 and on the full training dataset; thus, we will use it as primary on the test dataset for *add-1* and *add-2*.

For interpolation-based methods, we need to choose a value for the interpolation parameters. Due to time constraints, we use the same value for the phrase translation probabilities and for the lexicalized probabilities, and we perform grid search for $\alpha \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$ using *interpolate* on the full training dataset. As Table 5 shows, $\alpha = 0.6$ turns out to work best on the development dataset; we will use this value in our experiments on the test dataset both for *interpolate* and for *ourMethod*[8].

α	0.3	0.4	0.5	0.6	0.7
BLEU	14.17	14.49	14.6	14.73	14.52

Table 5: **Trying different values for *interpolate*** (on dev dataset). BLEU (in %) is for the full training dataset.

Evaluation on the test dataset. We integrate the morphologically enhanced system *m+phr+lm* and the word-token based *w-system* using the four merging methods above. The results for the full training dataset are shown in Table 6. As we can see, *add-1* and *add-2* make little difference compared to the *m-system* baseline. In contrast, *interpolation* and *ourMethod* yield sizable absolute improvements of 0.55 and 0.74 BLEU points, respectively, over the *m-system*; moreover, they outperform the *w-system*.

	Merging methods	Full (714K)
(i)	*m-system*	14.08
	w-system	14.58
(ii)	*add-1*	$14.25^{+0.17}$
	add-2	$13.89^{-0.19}$
(iii)	*interpolation*	$14.63^{+0.55}$
	ourMethod	**14.82**$^{+0.74}$

Table 6: **Merging *m+phr+lm* and *w-system*** (on test dataset). BLEU (in %) is for the full training dataset. Superscripts indicate performance gain/loss w.r.t *m-system*.

6 Discussion

Below we assess the significance of our results based on micro-analysis and human judgments.

[6] The feature values are e^1, $e^{2/3}$ or $e^{1/3}$ (e=2.71828...); when the phrase pair comes from both tables, from the primary table only, and from the secondary table only, respectively.

[7] The feature values are (e^1, e^1), (e^1, e^0) or (e^0, e^1) when the phrase pair comes from both tables, from the primary table only, and from the secondary table only, respectively.

[8] Note that this might put *ourMethod* at disadvantage.

6.1 Translation Model Comparison

We first compare the following three phrase tables: PT_m of *m-system*, maximum phrase length of 10 morpheme-tokens; $PT_{w \to m}$ of *w-system*, maximum phrase length of 7 word-tokens, re-segmented into morpheme-tokens; and PT_{m+phr} – morpheme-token input using word boundary-aware phrase extraction, maximum phrase length of 7 word-tokens.

		Full (714K)
(i)	PT_m	43.5M
	$PT_{w \to m}$	28.9M
	PT_{m+phr}	22.5M
(ii)	$PT_{m+phr} \bigcap PT_m$	21.4M
	$PT_{m+phr} \bigcap PT_{w \to m}$	10.7M

Table 7: **Phrase table statistics.** The number of phrase pairs in (i) individual PTs and (ii) PT overlap, is shown.

$\boldsymbol{PT_{m+phr}}$ **versus** $\boldsymbol{PT_m}$**.** Table 7 shows that PT_{m+phr} is about half the size of PT_m. Still, as Table 3 shows, *m+phr* outperforms the *m-system*. Moreover, 95.07% (21.4M/22.5M) of the phrase pairs in PT_{m+phr} are also in PT_m, which confirms that boundary-aware phrase extraction selects good phrase pairs from PT_m to be retained in PT_{m+phr}.

$\boldsymbol{PT_{m+phr}}$ **versus** $\boldsymbol{PT_{w \to m}}$**.** These two tables are comparable in size: 22.5M and 28.9M pairs, but their overlap is only 47.67% (10.7M/22.5M) of PT_{m+phr}. Thus, enriching the translation model with $PT_{w \to m}$ helps improve coverage.

6.2 Significance of the Results

Table 8 shows the performance of our system compared to the two baselines: *m-system* and *w-system*. We achieve an absolute improvement of 0.74 BLEU points over the *m-system*, from which our system evolved. This might look modest, but note that the baseline BLEU is only 14.08, and thus the relative improvement is 5.6%, which is not trivial. Furthermore, we outperform the *w-system* by 0.24 points (1.56% relative). Both improvements are statistically significant with $p < 0.01$, according to Collins' sign test (Collins et al., 2005).

In terms of m-BLEU, we achieve an improvement of 2.59 points over the *w-system*, which suggest our system might be performing better than what standard BLEU suggests. Below we test this hypothesis by means of micro-analysis and human evaluation.

	BLEU	**m-BLEU**
ourSystem	14.82	55.64
m-system	14.08	55.26
w-system	14.58	53.05

Table 8: **Our system vs. the two baselines** (on the test dataset): BLEU and m-BLEU scores (in %).

Translation Proximity Match. We performed automatic comparison based on corresponding phrases between the translation output (*out*) and the reference (*ref*), using the source (*src*) test dataset as a pivot. The decoding log gave us the phrases used to translate *src* to *out*, and we only needed to find correspondences between *src* and *ref*, which we accomplished by appending the test dataset to training and performing IBM Model 4 word alignments.

We then looked for phrase triples (*src*, *out*, *ref*), where there was a high character-level similarity between *out* and *ref*, measured using *longest common subsequence ratio* with a threshold of 0.7, set experimentally. We extracted 16,262 triples: for 6,758 of them, the translations matched the references exactly, while in the remaining triples, they were close wordforms[9]. These numbers support the hypothesis that our approach yields translations close to the reference wordforms but unjustly penalized by BLEU, which only gives credit for exact word matches[10].

Human Evaluation. We asked four native Finnish speakers to evaluate 50 random test sentences. Following (Callison-Burch et al., 2009), we provided them with the source sentence, its reference translation, and the outputs of three SMT systems (*m-system*, *w-system*, and *ourSystem*), which were shown in different order for each example and were named *sys1*, *sys2* and *sys3* (by order of appearance). We asked for three pairwise judgments: (i) *sys1* vs. *sys2*, (ii) *sys1* vs. *sys3*, and (iii) *sys2* vs. *sys3*. For each pair, a winner had to be designated; ties were allowed. The results are shown in Table 10. We can see that the judges consistently preferred

[9] Examples of such triples are (`constitutional structure`, perustuslaillinen rakenne, perustuslaillisempi rakenne) and (`economic and social`, taloudellisia ja sosiaalisia, taloudellisten ja sosiaalisten)

[10] As a reference, the *w-system* yielded 15,673 triples, and 6,392 of them were exact matches. Compared to our system, this means 589 triples and 366 exact matches less.

`src`: as a conservative , i am incredibly thrifty with taxpayers ’ money .
`ref`: maltillisen kokoomuspuolueen edustajana suhtaudun **erittain saastavaisesti veronmaksajien** rahoihin .
`our`: konservatiivinen , olen **erittain saastavaisesti veronmaksajien** rahoja .
`w`   : konservatiivinen , olen aarettoman tarkeaa kanssa *veronmaksajien* rahoja .
`m`   : *kuten* konservatiivinen , olen **erittain saastavaisesti veronmaksajien** rahoja .
Comment: **`our`** $\succ$ **`m`** $\succ$ **`w`**. **`our`** uses better paraphrases, from which the correct meaning could be inferred. The part “aarettoman tarkeaa kanssa” in **`w`** does not mention the “thriftiness” and replaces it with “important” (tarkeaa), which is wrong. **`m`** introduces “kuten”, which slightly alters the meaning towards “like a conservative, ...”.

`src`: we were very constructive and we negotiated until the last minute of these talks in the hague .
`ref`: olimme erittain **rakentavia** ja neuvottelimme haagissa **viime hetkeen saakka** .
`our`: olemme olleet hyvin **rakentavia** ja olemme neuvotelleet **viime hetkeen saakka** naiden neuvottelujen haagissa .
`w`   : olemme olleet hyvin **rakentavia** ja olemme neuvotelleet *viime tippaan niin* naiden neuvottelujen haagissa .
`m`   : olimme erittain *rakentavan* ja neuvottelimme **viime hetkeen saakka** naiden neuvotteluiden haagissa .
Comment: **`our`** $\succ$ **`m`** $\succeq$ **`w`**. In **`our`**, the meaning is very close to **`ref`** with only a minor difference in tense at the beginning. **`m`** only gets the case wrong in “rakentavan”, and the correct case is easily guessable. For **`w`**, the “viime tippaan” is in principle correct but somewhat colloquial, and the “niin” is extra and somewhat confusing.

`src`: it would be a very dangerous situation if the europeans were to become logistically reliant on russia .
`ref`: olisi **erittäin** vaarallinen tilanne , jos **eurooppalaiset** tulisivat **logistisesti** riippuvaisiksi venäjästä .
`our`: olisi **erittäin** vaarallinen tilanne , jos **eurooppalaiset** tulee **logistisesti** riippuvaisia venäjän .
`w`   : *se* olisi **erittäin** vaarallinen tilanne , jos *eurooppalaisten* tulisi *logistically* riippuvaisia venäjän .
`m`   : *se* olisi *hyvin* vaarallinen tilanne , jos **eurooppalaiset** *haluavat* tulla **logistisesti** riippuvaisia venäjän .
Comment: **`our`** $\succ$ **`w`** $\succeq$ **`m`**. **`our`** is almost correct except for the wrong inflections at the end. **`w`** is inferior since it failed to translate “logistically”. “haluavat tulla” in **`m`** suggests that the Europeans would “want to become logistically dependent”, which is not the case. The “se” (it), and “hyvin” (a synonym of “erittäin”) are minor mistakes/differences.

Table 9: **English-Finnish translation examples**. Shown are the source (`src`), the reference (`ref`), and the translations of three systems (`our`, `w`, `m`). Text in bold indicates matches with respect to the `ref`, while italics show where a system was judged inferior to the rest, as judged by native Finnish speakers.

(1) *ourSystem* to the *m-system*, (2) *ourSystem* to the *w-system*, (3) *w-system* to the *m-system*. These preferences are statistically significant, as found by the sign test. Comparing to Table 8, we can see that BLEU correlates with human judgments better than m-BLEU; we plan to investigate this in future work.

	our vs. m		**our vs. w**		**w vs. m**	
Judge 1	25	18	19	12	21	19
Judge 2	24	16	19	15	25	14
Judge 3	**27**†	**12**	17	11	**27**†	**15**
Judge 4	25	20	**26**†	**12**	22	22
Total	**101**‡	**66**	**81**‡	**50**	**95**†	**70**

Table 10: **Human judgments:** *ourSystem* (our) vs. *m-system* (m) vs. *w-system* (w). For each pair, we show the number of times each system was judged better than the other one, ignoring ties. Statistically significant differences are marked with † ($p < 0.05$) and ‡ ($p < 0.01$).

Finally, Table 9 shows some examples demonstrating how our system improves over the *w-system* and the *m-system*.

7 Conclusion and Future Work

In the quest towards a morphology-aware SMT that only uses unannotated data, there are two key challenges: (1) to bring the performance of morpheme-token systems to a level rivaling the standard word-token ones, and (2) to incorporate morphological analysis directly into the translation process.

This work satisfies the first challenge: we have achieved statistically significant improvements in BLEU for a large training dataset of 714K sentence pairs and this was confirmed by human evaluation.

We think we have built a solid framework for the second challenge, and we plan to extend it further.

Acknowledgements

We thank Joanna Bergström-Lehtovirta (Helsinki Institute for Information Technology), Katri Haverinen (University of Turku and Turku Centre for Computer Science), Veronika Laippala (University of Turku), and Sampo Pyysalo (University of Tokyo) for judging the Finnish translations.

References

Eleftherios Avramidis and Philipp Koehn. 2008. Enriching morphologically poor languages for statistical machine translation. In *ACL-HLT*.

Ibrahim Badr, Rabih Zbib, and James Glass. 2008. Segmentation for English-to-Arabic statistical machine translation. In *ACL-HLT*.

Tim Buckwalter. 2004. Buckwalter Arabic Morphological Analyzer Version 2.0. Linguistic Data Consortium, Philadelphia”.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. 2009. Findings of the 2009 Workshop on Statistical Machine Translation. In *EACL*.

Boxing Chen, Min Zhang, Haizhou Li, and Aiti Aw. 2009a. A comparative study of hypothesis alignment and its improvement for machine translation system combination. In *ACL-IJCNLP*.

Yu Chen, Michael Jellinghaus, Andreas Eisele, Yi Zhang, Sabine Hunsicker, Silke Theison, Christian Federmann, and Hans Uszkoreit. 2009b. Combining multi-engine translations with Moses. In *EACL*.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *ACL*.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Trans. Speech Lang. Process.*, 4(1):3.

Thi Ngoc Diep Do, Viet Bac Le, Brigitte Bigi, Laurent Besacier, and Eric Castelli. 2009. Mining a comparable text corpus for a Vietnamese-French statistical machine translation system. In *EACL*.

Sharon Goldwater and David McClosky. 2005. Improving statistical MT through morphological analysis. In *HLT*.

Philipp Koehn and Barry Haddow. 2009. Edinburgh’s submission to all tracks of the WMT2009 shared task with reordering and speed improvements to Moses. In *EACL*.

Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *EMNLP-CoNLL*.

Philipp Koehn and Christof Monz. 2005. Shared task: Statistical machine translation between European languages. In *WPT*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *NAACL*.

Philipp Koehn, Hieu Hoang, Alexandra Birch Mayne, Christopher 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 *ACL, Demonstration Session*.

Philipp Koehn. 2003. *Noun phrase translation*. Ph.D. thesis, University of Southern California, Los Angeles, CA, USA.

Young-Suk Lee. 2004. Morphological analysis for statistical machine translation. In *HLT-NAACL*.

Preslav Nakov and Hwee Tou Ng. 2009. Improved statistical machine translation for resource-poor languages using related resource-rich languages. In *EMNLP*.

Jan Niehues, Teresa Herrmann, Muntsin Kolss, and Alex Waibel. 2009. The Universität Karlsruhe translation system for the EACL-WMT 2009. In *EACL*.

Attila Novák. 2009. MorphoLogic’s submission for the WMT 2009 shared task. In *EACL*.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL*.

Kemal Oflazer and Ilknur El-Kahlout. 2007. Exploring different representational units in English-to-Turkish statistical machine translation. In *StatMT*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Fatiha Sadat and Nizar Habash. 2006. Combination of Arabic preprocessing schemes for statistical machine translation. In *ACL*.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*.

Kristina Toutanova, Hisami Suzuki, and Achim Ruopp. 2008. Applying morphology generation models to machine translation. In *ACL-HLT*.

Sami Virpioja, Jaakko J. Vyrynen, Mathias Creutz, and Markus Sadeniemi. 2007. Morphology-aware statistical machine translation based on morphs induced in an unsupervised manner. In *Machine Translation Summit XI*.

Hua Wu and Haifeng Wang. 2007. Pivot language approach for phrase-based statistical machine translation. *Machine Translation*, 21(3):165–181.

Mei Yang and Katrin Kirchhoff. 2006. Phrase-based backoff models for machine translation of highly inflected languages. In *EACL*.

“Poetic” Statistical Machine Translation: Rhyme and Meter

Dmitriy Genzel **Jakob Uszkoreit** **Franz Och**
Google, Inc.
1600 Amphitheatre Pkwy
Mountain View, CA 94043, USA
{dmitriy,uszkoreit,och}@google.com

Abstract

As a prerequisite to translation of poetry, we implement the ability to produce translations with meter and rhyme for phrase-based MT, examine whether the hypothesis space of such a system is flexible enough to accomodate such constraints, and investigate the impact of such constraints on translation quality.

1 Introduction

Machine translation of poetry is probably one of the hardest possible tasks that can be considered in computational linguistics, MT, or even AI in general. It is a task that most humans are not truly capable of. Robert Frost is reported to have said that poetry is that which gets lost in translation. Not surprisingly, given the task’s difficulty, we are not aware of any work in the field that attempts to solve this problem, or even discuss it, except to mention its difficulty, and professional translators like to cite it as an example of an area where MT will never replace a human translator. This may well be true in the near or even long term. However, there are aspects of the problem that we can already tackle, namely the problem of the poetic form.

Vladimir Nabokov, in his famous translation of *Eugene Onegin* (Nabokov, 1965), a poem with a very strict meter and rhyming scheme, heavily disparages those translators that attempt to preserve the form, claiming that since it is impossible to perfectly preserve both the form and the meaning, the form must be entirely sacrificed. On the other hand, Douglas Hofstadter, who spends 600 pages describing how to translate a 60 word poem in 80 different ways in *Le Ton beau de Marot* (1998), makes a strong case that a poem’s form must be preserved in translation, if at all possible. Leaving the controversy to the professional translators, we investigate whether or not it is possible to produce translations that conform to certain metrical constraints common in poetry.

Statistical machine translation techniques, unlike their traditional rule-based counterparts, are in fact well-suited to the task. Because the number of potential translation hypotheses is very large, it is not unreasonable to expect that some of them should conform to an externally imposed standard. The goal of this paper is to investigate how these hypotheses can be efficiently identified, how often they are present, and what the quality penalty for imposing them is.

2 Related Work

There has been very little work related to the translation of poetry. There has been some work where MT techniques were used to produce poetry (Jiang and Zhou, 2008). In other computational poetry work, Ramakrishnan et al (2009) generate song lyrics from melody and various algorithms for poetry generation (Manurung et al., 2000; Díaz-Agudo et al., 2002). There are books (Hartman, 1996) and articles (Bootz, 1996) on the subject of computer poetry from a literary point of view. Finally, we must mention Donald Knuth’s seminal work on complexity of songs (Knuth, 1984).

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 158–166,
MIT, Massachusetts, USA, 9-11 October 2010.

3 Statistical MT and Poetry

We can treat any poetic form as a constraint on the potential outputs. A naive approach to ensure that an output of the MT system is, say, a *haiku*, is to create a *haiku* detector and to examine a (very large) n-best list of translations. This approach would not succeed very often, since the *haikus* that may be among the possible translations are a very small fraction of all translations, and the MT decoder is not actively looking for them, since it is not part of the cost it attempts to minimize. Instead, we would want to recast "Haikuness" as a feature function, such that a real *haiku* has 0 cost, and those outputs that are not, have large cost. This feature function must be local, rather than global, so as to guide the decoder search.

The concept of feature functions as used in statistical MT is described by Och and Ney (Och and Ney, 2002). For a phrase based system, a feature function is a function whose inputs are a partial hypothesis state s_{in}, and a phrase pair p, and whose outputs are the hypothesis state after p is appended to the hypothesis: s_{out}, and the cost incurred, c. For hierarchical, tree-to-string and some other types of MT systems which combine two partial hypotheses and are not generating translations left-to-right, one instead has two partial hypotheses states s_{left} and s_{right} as inputs, and the outputs are the same. Our first goal is to describe how these functions can be efficiently implemented.

The feature function costs are multiplied by fixed weights and added together to obtain the total hypothesis cost. Normally feature functions include the logarithm of probability of target phrase given source, source given target and other phrase-local features which require no state to be kept, as well as features like language model, which require non-trivial state. The weights are usually learned automatically, however we will set them manually for our feature functions to be effectively infinite, since we want them to override all other sources of information.

We will now examine some different kinds of poetry and consider the properties of such feature functions, especially with regard to keeping necessary state. We are concerned with minimizing the amount of information to be kept, both due to memory requirements, and especially to ensure that compatible hypotheses can be efficiently recombined by the decoder.

3.1 Line-length constrained poetry

Some poetic genres, like the above-mentioned *haiku*, require that a poem contain a certain number of lines (3 for *haiku*), each containing a certain number of syllables (5,7,5 for *haiku*). These genres include *lanternes*, *fibs*, *tankas*, and many others. These genres impose two constraints. The first constraint is on total length. This requires that each hypothesis state contain the current translation length (in syllables). In addition, whenever a hypothesis is expanded, we must keep track of whether or not it would be possible to achieve the desired final length with such an expansion. For example, if in the initial state, we have a choice between two phrases, and picking the longer of the two would make it impossible to have a 17-syllable translation later on, we must impose a high cost on it, so as to avoid going down a garden path.

The second constraint is on placing line breaks: they must come at word boundaries. Therefore the 5th and 12th (and obviously 17th) syllable must end words. This also requires knowing the current hypothesis' syllable length, but unlike the first constraint, it can be scored entirely locally, without considering possible future expansions. For either constraint, however, the sentence has to be assembled strictly left-to-right, which makes it impossible to build partial hypotheses that do not start the sentence, which hierarchical and tree-to-string decoders require.

3.2 Rhythmic poetry

Some famous Western poetry, notably Shakespeare, is written in rhythmic poetry, also known as *blank verse*. This poetry imposes a constraint on the pattern of stressed and unstressed syllables. For example, if we use 0 to indicate no stress, and 1 to indicate stress, *blank verse* with iambic foot obeys the regular expression $(01)*$, while one with a dactylic foot looks like $(100)*$. This genre is the easiest to handle, because it does not require current position, but only its value modulo foot length (e.g. for an iamb, whether the offset is even or odd). It is even possible, as described in Section 4, to track this form in a decoder that is not left-to-right.

3.3 Rhythmic and rhyming poetry

The majority of English poetry that was written until recently has both rhythm and rhyme. Generally speaking, a poetic genre of this form can be described by two properties. The first is a rhyming scheme. A rhyming scheme is a string of letters, each corresponding to a line of a poem, such that the same letter is used for the lines that rhyme. E.g. *aa* is a scheme for a couplet, a 2-line poem whose lines rhyme. A *sonnet* might have a complicated scheme like *abbaabbacdecde*. The second property concerns meter. Usually lines that rhyme have the same meter (i.e. the exact sequence of stressed and unstressed syllables). For example, an *iambic pentameter* is an *iamb* repeated 5 times, i.e. *0101010101*. We can describe a genre completely by its rhyming scheme and a meter for each letter used in the rhyming scheme. We will refer to this object as *genre description*. E.g. $\{abab, a : 010101, b : 10101010\}$ is a quatrain with trimeter iambic and tetrameter trochaic lines. Note that the other two kinds of poetry can also be fit by this structure, if one permits another symbol (we use *) to stand for the syllables whose stress is not important, e.g. a *haiku*: $\{abc, a : *****, b : *******, c : *****\}$. For this type of genre, we need to obey the same two constraints as in the line-based poetry, but also to ensure that rhyming constraints hold. This requires us to include in a state, for any outstanding rhyme letter, the word that occurred at the end of that line. It is not sufficient to include just the syllable that must rhyme, because we wish to avoid self-rhymes (word rhyming with an identical word).

4 Stress pattern feature function

We will first discuss an easier special case, namely a feature function for blank verse, which we will refer to as *stress pattern* feature function. This feature function can be used for both phrase-based and hierarchical systems.

In addition to a statistical MT system (Och and Ney, 2004; Koehn et al., 2007), it is necessary to have the means to count the syllables in a word and to find out which ones are stressed. This can be done with a pronunciation module of a text-to-speech system, or a freely available pronunciation dictionary, such as CMUDict (Rudnicky, 2010). Out-of-vocabulary words can be treated as always imposing a high cost.

4.1 Stress pattern for a phrase-based system

In a phrase based system, the feature function state consists of the current hypothesis length modulo foot length. For a 2-syllable foot, it is either 0 or 1, for a 3-syllable foot, 0, 1, or 2. The proposed target phrase is converted into a stress pattern using the pronunciation module, and the desired stress pattern is left shifted by the current offset. The cost is the number of mismatches of the target phrase vs. the pattern. For example, if the desired pattern is *010*, current offset is 1, and the proposed new phrase has pattern *10011*, we shift the desired pattern by 1, obtaining *100* and extend it to length 5, obtaining *10010*, matching it against the proposal. There is one mismatch, at the fifth position, and we report a cost of 1. The new state is simply the old state plus phrase length, modulo foot length, 0 in this example.

4.2 Stress pattern for a hierarchical system

In a hierarchical system, we in general do not know how a partial hypothesis might be combined on the left. A hypothesis that is a perfect fit for pattern *010* would be horrible if it is placed at an offset that is not divisible by 3, and vice versa, an apparently bad hypothesis might be perfectly good if placed at such an offset. To solve this problem, we create states that track how well a partial hypothesis fits not only the desired pattern, but all patterns obtained by placing this pattern at any offset, and also the hypothesis length (modulo foot length, as usual). For instance, if we observe a pattern *10101*, we record the following state: {*length: 1, 01 cost: 5, 10 cost: 0*}. If we now combine this state with another, such as {*length: 0, 01 cost: 1, 10 cost: 0*}, we simply add the lengths, and combine the costs either of the same kind (if left state's length is even), or the opposite (if it is odd). In this instance we get {*length: 1, 01 cost: 5, 10 cost: 1*}. If both costs are greater than 0, we can subtract the minimum cost and immediately output it as cost: this is the unavoidable cost of this combination. For this example we get cost of 1, and a new state: {*length: 1, 01 cost: 4, 10 cost: 0*}. For the final state, we output the remaining cost for the pattern we desire. The approach is very similar for feet of length 3.

4.3 Stress pattern: Whatever fits

With a trivial modification we can output translations that can fit any one of the patterns, as long as we do not care which. The approach is identical for both hierarchical and phrase-based systems. We simply track all foot patterns (length 2 and length 3 are the only ones used in poetry) as in the above algorithm, taking care to combine the right pattern scores based on length offset. The length offset now has to be tracked modulo 2*3.

This feature function can now be used to translate arbitrary text into blank verse, picking whatever meter fits best. If no meters can fit completely, it will produce translations with the fewest violations (assuming the weight for this feature function is set high).

5 General poetic form feature function

In this section we discuss a framework for tracking any poetic genre, specified as a genre description object (Section 3.3 above). As in the case of the stress pattern function, we use a statistical MT system, which is now required to be phrase-based only. We also use a pronunciation dictionary, but in addition to tracking the number and stress of syllables, we must now be able to provide a function that classifies a pair of words as rhyming or non-rhyming. This is in itself a non-trivial task (Byrd and Chodorow, 1985), due to lack of a clear definition of what constitutes a rhyme. In fact rhyming is a continuum, from very strong rhymes to weak ones. We use a very weak definition which is limited to a single syllable: if the final syllables of both words have the same nucleus and coda[1], we say that the words rhyme. We accept this weak definition because we prefer to err on the side of over-generation and accept even really bad poetry.

5.1 Tracking the target length

The hardest constraint to track efficiently is the range of lengths of the resulting sentence. Phrase-based decoders use a limited-width beam as they build up possible translations. Once a hypothesis drops out of the beam, it cannot be recovered, since no backtracking is done. Therefore we cannot afford to explore a part of the hypothesis space which has no possible solutions for our constraints, we must be able to prune a hypothesis as soon as it leads us to such a subspace, otherwise we will end up on an unrecoverable garden path. To avoid this problem, we need to have a set of possible sentence lengths available at any point in the search, and to impose a high cost if the desired length is not in that set.

Computing this set exactly involves a standard dynamic programming sweep over the phrase lattice, including only uncovered source spans. If the maximum source phrase size is k, source sentence length is n and maximum target/source length ratio for a phrase is l (and therefore target sentence is limited to at most ln words), this sweep requires going over $O(n^2)$ source ranges, each of which can be produced in k ways, and tracking ln potential lengths in each, resulting in $O(n^3kl)$ algorithm. This is unacceptably slow to be done for each hypothesis (even noting that hypotheses with the same set of already covered source position can share this computation).

We will therefore solve this task approximately. First, we can note that in most cases the set of possible target lengths is a range. This is due to phrase extraction constraints, which normally ensure that the lengths of target phrases form a complete range. This means that it is sufficient to track only a minimum and maximum value for each range, reducing time to $O(n^2k)$. Second, we can note that whenever a source range is interrupted by a covered phrase and split into two ranges, the minimal and maximal sentence length is simply the sum of the corresponding lengths over the two uncovered subranges, plus the current hypothesis length. Therefore, if we precompute the minimum and maximum lengths over all ranges, using the same dynamic programming algorithm in advance, it is only necessary to iterate over the uncovered ranges (at most $O(n)$, and $O(1)$ in practice, due to reordering constraints) at runtime and sum their minimum and maximum values. As a result, we only need to do $O(n^2k)$ work upfront, and on average $O(1)$ extra work for each hypothesis.

5.2 State space

A state for the feature function must contain the following elements:

- Current sentence length (in syllables)

[1] In phonology, nucleus and coda together are in fact called *rhyme* or *rime*

- Set of uncovered ranges (as needed for the computation above)

- Zero or more letters from the rhyming scheme with the associated word that has an outstanding rhyme

5.3 The combination algorithm

To combine the hypothesis state s_{in} with a phrase pair p, do the following

1. Initialize cost as 0, s_{out} as s_{in}

2. Update s_{out}: increment sentence length by target phrase length (in syllables), update coverage range

3. Compute minimum and maximum achievable sentence length; if desired length not in range, increment cost by a penalty

4. For each word in the target phrase

   (a) If the word's syllable pattern does not match against desired pattern, add number of mismatches to cost

   (b) If at the end of a line:

      i. If the line would end mid-word, increment cost by a penalty

      ii. Let x be this line's rhyme scheme letter

      iii. If x is present in the state s_{out}, check if the word associated with x rhymes with the current word, if not, increment cost by a penalty

      iv. Remove x with associated word from the state s_{out}

      v. If letter x occurs further in the rhyming scheme, add x with the current word to the state s_{out}

5.4 Tracking multiple patterns

The above algorithm will allow to efficiently search the hypothesis space for a single *genre description* object. In practice, however, there may be several desirable patterns, any one of which would be acceptable. A naive approach, to use multiple feature functions, one with each pattern, does not work, since the decoder is using a (log-)linear model, in which costs are additive. As a result, a pattern that matches one pattern, but not another, will still have high cost, perhaps as high as a pattern that partially matches both. We need to combine feature functions not linearly, but with a *min* operator. This is easily achieved by creating a combined state that encodes the union of each individual function's states (which can share most of the information), and in addition each feature function's current total cost. As long as at least one function has zero cost (i.e. can potentially match), no cost is reported to the decoder. As soon as all costs become positive, the minimum over all costs is reported to the decoder as unavoidable cost, and should be subtracted from each feature function cost, bringing the minimum stored in the output state back to 0.

It is also possible to prune the set of functions that are still viable, based on their cost, to avoid keeping track of patterns that cannot possibly match. Using this approach we can translate arbitrary text, provide a large number of poetic patterns and expect to get some sort of poem at the end. Given a wide variety of poetic genres, it is not unreasonable to expect that for most inputs, some pattern will apply. Of course, for translating actual poetry, we would likely have a specific form in mind, and a positive outcome would be less likely.

6 Results

We train a baseline phrase-based French-English system using WMT-09 corpora (Callison-Burch et al., 2009) for training and evaluation. We use a proprietary pronunciation module to provide phonetic representation of English words.

6.1 Stress Pattern Feature Function

We have no objective means of "poetic" quality evaluation. We are instead interested in two metrics: percentage of sentences that can be translated while obeying a stress pattern constraint, and the impact of this constraint on BLEU score (Papineni et al., 2002). Obviously, WMT test set is not itself in any way poetic, so we use it merely to see if arbitrary text can be forced into this constraint.

The BLEU score impact on WMT has been fairly consistent during our experimentation: the BLEU score is roughly halved. In particular, for the above system the baseline score is **35.33**, and stress

Table 1: Stress pattern distribution

Name	Pattern	% of matches
Iamb	01	9.6%
Trochee	10	7.2%
Anapest	001	27.1%
Amphibrach	010	32.1%
Dactyl	100	23.8%

pattern-constrained system only obtains **18.93**.

The proportion of sentences successfully matched is **85%**, and if we permit a single stress error, it is **93%**, which suggests that the constraint can be satisfied in the great majority of cases. The distribution of stress patterns among the perfect matches is given in Table 1.

Some of the more interesting example translations with stress pattern enforcement enabled are given in table 2.

6.2 Poetic Form Feature Function

For poetic form feature function, we perform the same evaluation as above, to estimate the impact of forcing prose into an arbitrary poetic form, but to get more relevant results we also translate a poetic work with a specific genre requirement.

Our poetic form feature function is given a list of some 210 genre descriptions which vary from Haikus to Shakespearean sonnets. Matching any one of them satisfies the constraint. We translate WMT blind set and obtain a BLEU score of **17.28** with the baseline of **35.33** as above. The proportion of sentences that satisfied one of the poetic constraints is **87%**. The distribution of matched genres is given in Table 3. Some of the more interesting example translations are given in table 2.

For a proper poetic evaluation, we use a French translation of Oscar Wilde's *Ballad of Reading Gaol* by Jean Guiloineau as input, and the original Wilde's text as reference. The poem consists of 109 stanzas of 6 lines each, with a genre description of {*abcbdb, a/c/d: 01010101, b: 010101*}. The French version obeys the same constraint. We treat each stanza as a sentence to be translated. The baseline BLEU score is **10.27**. This baseline score is quite low, as can be expected for matching a literal MT translation against a professional poetical translation. We evaluate our system with a poetic constraint given above.

Table 3: Genre distribution for WMT corpus.
(Descriptions of these genres can be found in Wikipedia, http://en.wikipedia.org)

Genre	Number	Percentage
No poem	809	13.1%
Blank verse	5107	82.7%
Couplet	81	1.3%
Haiku	42	0.7%
Cinquain	33	0.5%
Dodoitsu	24	0.4%
Quinzaine	23	0.4%
Choka	18	0.3%
Fib	15	0.2%
Tanka	14	0.2%
Lanterne	4	0.1%
Triplet	1	0.02%
Quatrain	1	0.02%
Total	6172	100%

The resulting score is **7.28**. Out of 109 stanzas, we found 12 translations that satisfy the genre constraint (If we allow any poetic form, 108 out of 109 stanzas match some form). Two sample stanzas that satisfied the constraints are given in Table 4.

7 Discussion and Future Work

In this work we demonstrate how modern-day statistical MT system can be constrained to search for translations obeying particular length, meter, and rhyming constraints, whether a single constraint, or any one of a set. We further demonstrate that the hypothesis space is often rich enough that these constraints can be satisfied. The impact on translation quality, however, is quite profound, as is to be expected. It seems that at the present state of machine translation, one does indeed have to choose between getting either the form or the meaning right. In the present form, however, we can already find good translations, as a sort of *found poetry* (Drury, 2006), by translating a large quantity of text, whether poetic or not.

This is the first attempt to deal with poetry translation, and the great majority of work to achieve reasonable quality in form and meaning still remains to be done. One major problem with the current feature function is that while it can avoid the areas of the search space where length constraints cannot be

Table 2: Example translations. Stressed syllables are italicized

Reference	A police spokesman said three people had been arrested and the material was being examined.
Baseline	A policeman said that three people were arrested and that the material is currently being analyzed.
Stress Pattern (001)	A po*lice* said that *three* were ar*rest*ed and *that* the e*quip*ment is *cur*rently *be*ing e*xa*mined.
Poetic: *Couplet in amphibrachic tetrameter*	An *of*ficer *sta*ted that *three* were ar*rest*ed and *that* the e*quip*ment is *cur*rently *test*ed.

Reference	A trio of retired generals launched a mutiny in the Lords, protesting against cuts in military spending: being armed-forces minister is, they claimed, a part-time job.
Baseline	A trio of retired generals have launched a revolt among the Lords, protesting against cuts in military spending: they have proclaimed only Minister of Defence is for them, a part-time employment.
Stress Pattern (010)	A *tri*o of *ge*neral re*ti*rement *launched* a re*bel*lion a*mong* Lords, pro*test*ing the *spen*ding cuts *troops*: they claimed *Mi*nister *on*ly de*fen*se is *for* them, a *part*-time job.
Poetic: *Blank Verse in amphibrachic trimeter*	A *tri*o of *ge*nerals re*ti*red have *launched* an up*ri*sing a*mong* Lords, pro*test*ing the *spend*ing cuts *mem*bers: they *mi*nister *on*ly pro*claimed* the de*fense* is for *them*, a part-*time* job.

Reference	We must continue to condemn human rights abuses in Burma.
Baseline	We must continue to denounce the violations of human rights abuses in Burma.
Stress Pattern (100)	*We* must con*ti*nue to *speak* out a*gainst* rights a*bu*ses com*mi*tted in *Bur*ma.
Poetic: *Haiku: 5-7-5 syllables*	We must continue denounce violations of human rights Burma.

Table 4: Sample translations from Oscar Wilde's *Ballad of Reading Gaol*.

Wilde's original	**Our translation**
He did not wring his hands, as do	Without hands twisted like these men,
Those witless men who dare	Poor men without hope, dare
To try to rear the changeling Hope	To nourish hope in our vault
In the cave of black Despair:	Of desperation there
He only looked upon the sun,	And looked toward the sun, drink cool
And drank the morning air.	Until the evening air.
With slouch and swing around the ring	We are in our circle we
We trod the Fool's Parade!	Dragged like the Fools' Parade!
We did not care: we knew we were	It mattered little, since we were
The Devil's Own Brigade:	The Devil's sad Brigade:
And shaven head and feet of lead	A shaved head and the feet of lead
Make a merry masquerade.	Regardless gay charade!

satisfied, it cannot avoid the areas where rhyming constraints are impossible to satisfy. As a result, we need to allow a very wide hypothesis beam (5000 per each source phrase coverage), to ensure that enough hypotheses are considered, so that there are some that lead to correct solutions later. We do not currently have a way to ensure that this happens, although we can attempt to constrain the words that end lines to have possible rhymes, or employ other heuristics. A more radical solution is to create an entirely different decoding algorithm which places words not left-to-right, or in a hierarchical fashion, but first placing words that must rhyme, and building hypotheses around them, like human translators of poetry do.

As a result, the system at present is too slow, and we cannot make it available online as a demo, although we may be able to do so in the future.

The current approach relies on having enough lexical variety in the phrase table to satisfy constraints. Since our goal is not to be literal, but to obtain a satisfactory compromise between form and meaning, it would clearly be beneficial to augment target phrases with synonyms and paraphrases, or to allow for words to be dropped or added.

8 Acknowledgements

We would like to thank all the members of the MT team at Google, especially Richard Zens and Moshe Dubiner, for their help. We are thankful to the anonymous reviewers for their comments, especially to the one that to our amazement did the entire review in verse[2].

References

P. Bootz. 1996. Poetic machinations. *Visible Language*, 30(2):118–37.

Roy J. Byrd and Martin S. Chodorow. 1985. Using an on-line dictionary to find rhyming words and pronunciations for unknown words. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, pages 277–283, Chicago, Illinois, USA, July. Association for Computational Linguistics.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. 2009. Findings of the 2009 Workshop on Statistical Machine Translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 1–28, Athens, Greece, March. Association for Computational Linguistics.

B. Díaz-Agudo, P. Gervás, and P. González-Calero. 2002. Poetry generation in colibriza. In *Advances in Case-Based Reasoning*, pages 157–159. Springer.

John Drury. 2006. *The poetry dictionary*. Writer's Digest Books.

C.O. Hartman. 1996. *Virtual muse: experiments in computer poetry*. Wesleyan University Press.

Douglas R. Hofstadter. 1998. *Le Ton Beau De Marot: In Praise of the Music of Language*. Perseus Books Group.

[2] With the reviewer's permission, we feel that the extra work done by the reviewer deserves to be seen by more than a few people, and make it available for you to view at: `http://research.google.com/archive/papers/review_in_verse.html`

Long Jiang and Ming Zhou. 2008. Generating Chinese couplets using a statistical MT approach. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 377–384, Manchester, UK, August. Coling 2008 Organizing Committee.

D.E. Knuth. 1984. The complexity of songs. *Communications of the ACM*, 27(4):344–346.

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 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, June. Association for Computational Linguistics.

H.M. Manurung, G. Ritchie, and H. Thompson. 2000. Towards a computational model of poetry generation. In *Proceedings of AISB Symposium on Creative and Cultural Aspects and Applications of AI and Cognitive Science*, pages 79–86. Citeseer.

Vladimir Nabokov. 1965. *Eugene Onegin: A Novel in Verse by Alexandr Pushkin, Translated from the Russian*. Bollingen Foundation.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Ananth Ramakrishnan A., Sankar Kuppan, and Sobha Lalitha Devi. 2009. Automatic generation of Tamil lyrics for melodies. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, pages 40–46, Boulder, Colorado, June. Association for Computational Linguistics.

Alex Rudnicky. 2010. The Carnegie Mellon pronouncing dictionary, version 0.7a. Online.

Efficient Graph-Based Semi-Supervised Learning of Structured Tagging Models

Amarnag Subramanya
Google Research
Mountain View, CA 94043
asubram@google.com

Slav Petrov
Google Research
New York, NY 10011
slav@google.com

Fernando Pereira
Google Research
Mountain View, CA 94043
pereira@google.com

Abstract

We describe a new scalable algorithm for semi-supervised training of conditional random fields (CRF) and its application to part-of-speech (POS) tagging. The algorithm uses a similarity graph to encourage similar n-grams to have similar POS tags. We demonstrate the efficacy of our approach on a domain adaptation task, where we assume that we have access to large amounts of unlabeled data from the target domain, but no additional labeled data. The similarity graph is used during training to smooth the state posteriors on the target domain. Standard inference can be used at test time. Our approach is able to scale to very large problems and yields significantly improved target domain accuracy.

1 Introduction

Semi-supervised learning (SSL) is the use of small amounts of labeled data with relatively large amounts of unlabeled data to train predictors. In some cases, the labeled data can be sufficient to provide reasonable accuracy on in-domain data, but performance on even closely related out-of-domain data may lag far behind. Annotating training data for all sub-domains of a varied domain such as all of Web text is impractical, giving impetus to the development of SSL techniques that can learn from unlabeled data to perform well across domains. The earliest SSL algorithm is self-training (Scudder, 1965), where one makes use of a previously trained model to annotate unlabeled data which is then used to re-train the model. While self-training is widely used and can yield good results in some applications (Yarowsky, 1995), it has no theoretical guarantees except under certain stringent conditions, which rarely hold in practice(Haffari and Sarkar, 2007).

Other SSL methods include co-training (Blum and Mitchell, 1998), transductive support vector machines (SVMs) (Joachims, 1999), and graph-based SSL (Zhu et al., 2003). Several surveys cover a broad range of methods (Seeger, 2000; Zhu, 2005; Chapelle et al., 2007; Blitzer and Zhu, 2008). A majority of SSL algorithms are computationally expensive; for example, solving a transductive SVM exactly is intractable. Thus we have a conflict between wanting to use SSL with large unlabeled data sets for best accuracy, but being unable to do so because of computational complexity. Some researchers attempted to resolve this conflict by resorting to approximations (Collobert et al., 2006), but those lead to suboptimal results (Chapelle et al., 2007).

Graph-based SSL algorithms (Zhu et al., 2003; Joachims, 2003; Corduneanu and Jaakkola, 2003; Belkin et al., 2005; Subramanya and Bilmes, 2009) are an important subclass of SSL techniques that have received much attention in the recent past, as they outperform other approaches and also scale easily to large problems. Here one assumes that the data (both labeled and unlabeled) is represented by vertices in a graph. Graph edges link vertices that are likely to have the same label. Edge weights govern how strongly the labels of the nodes linked by the edge should agree.

Most previous work in SSL has focused on unstructured classification problems, that is, problems with a relatively small set of atomic labels. There

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 167–176,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

has been much less work on SSL for structured prediction where labels are composites of many atomic labels with constraints between them. While the number of atomic labels might be small, there will generally be exponentially many ways to combine them into the final structured label. Structured prediction problems over sequences appear for example in speech recognition, named-entity recognition, and part-of-speech tagging; in machine translation and syntactic parsing, the output may be tree-structured.

Altun et al. (2005) proposed a max-margin objective for semi-supervised learning over structured spaces. Their objective is similar to that of manifold regularization (Belkin et al., 2005) and they make use of a graph as a smoothness regularizer. However their solution involves inverting a matrix whose size depends on problem size, making it impractical for very large problems. Brefeld and Scheffer (2006) present a modified version of the co-training algorithm for structured output spaces. In both of the above cases, the underlying model is based on structured SVM, which does not scale well to very large datasets. More recently Wang et al. (2009) proposed to train a conditional random field (CRF) (Lafferty et al., 2001) using an entropy-based regularizer. Their approach is similar to the entropy minimization algorithm (Grandvalet and Bengio, 2005). The problem here is that their objective is not convex and thus can pose issues for large problems. Further, graph-based SSL algorithms outperform algorithms based on entropy minimization (Chapelle et al., 2007).

In this work, we propose a graph-based SSL method for CRFs that is computationally practical for very large problems, unlike the methods in the studies cited above. Our method is scalable because it trains with efficient standard building blocks for CRF inference and learning and also standard graph label propagation machinery. Graph regularizer computations are only used for training, so at test time, standard CRF inference can be used, unlike in graph-based transductive methods. Briefly, our approach starts by training a CRF on the source domain labeled data, and then uses it to decode unlabeled data from the target domain. The state posteriors on the target domain are then smoothed using the graph regularizer. Best state sequences for the unlabeled target data are then created by Viterbi decoding with the smoothed state posteriors, and this automatic target domain annotation is combined with the labeled source domain data to retrain the CRF.

We demonstrate our new method in domain adaptation for a CRF part-of-speech (POS) tagger. While POS tagging accuracies have reached the level of inter-annotator agreement (>97%) on the standard PennTreebank test set (Toutanova et al., 2003; Shen et al., 2007), performance on out-of-domain data is often well below 90%, impairing language processing tasks that need syntactic information. For example, on the question domain used in this paper, the tagging accuracy of a supervised CRF is only 84%. Our domain adaptation algorithm improves performance to 87%, which is still far below in-domain performance, but a significant reduction in error.

2 Supervised CRF

We assume that we have a set of labeled source domain examples $\mathcal{D}_l = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{l}$, but only unlabeled target domain examples $\mathcal{D}_u = \{\mathbf{x}_i\}_{i=l+1}^{l+u}$. Here $\mathbf{x}_i = x_i^{(1)} x_i^{(2)} \cdots x_i^{(|\mathbf{x}_i|)}$ is the sequence of words in sentence i and $\mathbf{y}_i = y_i^{(1)} y_i^{(2)} \cdots y_i^{(|\mathbf{x}_i|)}$ is the corresponding POS tag sequence, with $y_i^{(j)} \in \mathcal{Y}$ where $\mathcal{Y}$ is the set of POS tags. Our goal is to learn a CRF of the form:

$$\mathrm{p}(\mathbf{y}_i|\mathbf{x}_i;\Lambda) \propto \exp\left(\sum_{j=1}^{N_i}\sum_{k=1}^{K}\lambda_k f_k(y_i^{(j-1)}, y_i^{(j)}, \mathbf{x}_i, j)\right)$$

for the target domain. In the above equation, $\Lambda = \{\lambda_1, \ldots, \lambda_K\} \in \mathbb{R}^K$, $f_k(y_i^{(j-1)}, y_i^{(j)}, \mathbf{x}_i, j)$ is the k-th feature function applied to two consecutive CRF states and some window of the input sequence, and λ_k is the weight of that feature. We discuss our features in detail in Section 6. Given only labeled data $\mathcal{D}_l$, the optimal feature weights are given by:

$$\Lambda^* = \underset{\Lambda \in \mathbb{R}^K}{\mathrm{argmin}} \left[-\sum_{i=1}^{l} \log \mathrm{p}(\mathbf{y}_i|\mathbf{x}_i;\Lambda) + \gamma\|\Lambda\|^2\right] \quad (1)$$

Here $\|\Lambda\|^2$ is the squared ℓ_2-norm and acts as the regularizer, and γ is a trade-off parameter whose setting we discuss in Section 6. In our case, we also have access to the unlabeled data $\mathcal{D}_u$ from the target domain which we would like to use for training the CRF. We first describe how we construct a similarity graph over the unlabeled which will be used in our algorithm as a graph regularizer.

3 Graph Construction

Graph construction is the most important step in graph-based SSL. The standard approach for unstructured problems is to construct a graph whose vertices are labeled and unlabeled examples, and whose weighted edges encode the degree to which the examples they link should have the same label (Zhu et al., 2003). Then the main graph construction choice is what similarity function to use for the weighted edges between examples. However, in structured problems the situation is more complicated. Consider the case of sequence tagging we are studying. While we might be able to choose some appropriate sequence similarity to construct the graph, such as edit distance or a string kernel, it is not clear how to use whole sequence similarity to constrain whole tag sequences assigned to linked examples in the learning algorithm. Altun et al. (2005) had the nice insight of doing the graph construction not for complete structured examples but instead for the *parts* of structured examples (also known as factors in graphical model terminology), which encode the local dependencies between input data and output labels in the structured problem. However, their approach is too demanding computationally (see Section 5), so instead we use local sequence contexts as graph vertices, exploting the empirical observation that the part of speech of a word occurrence is mostly determined by its local context.

Specifically, the set V of graph vertices consists of all the word n-grams[1] (*types*) that have occurrences (*tokens*) in training sentences (labeled and unlabeled). We partition $V = V_l \cup V_u$ where V_l corresponds to n-grams that occur at least once in the labeled data, and V_u corresponds to n-grams that occur only in the unlabeled data.

Given a symmetric similarity function between types to be defined below, we link types u and v with an edge of weight w_{uv}, defined as:

Description	Feature
Trigram + Context	$x_1\ x_2\ x_3\ x_4\ x_5$
Trigram	$x_2\ x_3\ x_4$
Left Context	$x_1\ x_2$
Right Context	$x_4\ x_5$
Center Word	x_2
Trigram – Center Word	$x_2\ x_4$
Left Word + Right Context	$x_2\ x_4\ x_5$
Left Context + Right Word	$x_1\ x_2\ x_4$
Suffix	HasSuffix(x_3)

Table 1: Features we extract given a sequence of words "$x_1\ x_2\ x_3\ x_4\ x_5$" where the trigram is "$x_2\ x_3\ x_4$".

$$w_{uv} = \begin{cases} \mathrm{sim}(u,v) & \text{if } v \in \mathcal{K}(u) \text{ or } u \in \mathcal{K}(v) \\ 0 & \text{otherwise} \end{cases}$$

where $\mathcal{K}(u)$ is the set of k-nearest neighbors of u according to the given similarity. For all experiments in this paper, $n = 3$ and $k = 5$.

To define the similarity function, for each token of a given type in the labeled and unlabeled data, we extract a set of context features. For example, for the token $x_2\ x_3\ x_4$ occurring in the sequence $x_1\ x_2\ x_3\ x_4\ x_5$, we use feature templates that capture the left ($x_1\ x_2$) and right contexts ($x_4\ x_5$). Additionally, we extract suffix features from the word in the middle. Table 1 gives an overview of the features that we used. For each n-gram type, we compute the vector of pointwise mutual information (PMI) values between the type and each of the features that occur with tokens of that type. Finally, we use the cosine distance between those PMI vectors as our similarity function.

We have thus circumvented the problem of defining similarities over sequences by defining the graph over types that represent local sequence contexts. Since our CRF tagger only uses local features of the input to score tag pairs, we believe that the graph we construct captures all significant context information. Figure 1 shows an excerpt from our graph. The figure shows the neighborhoods of a subset of the vertices with the center word 'book.' To reduce clutter, we included only closest neighbors and the edges that involve the nodes of interest.

[1] We pad the n-grams at the beginning and end of sentences with appropriate dummy symbols.

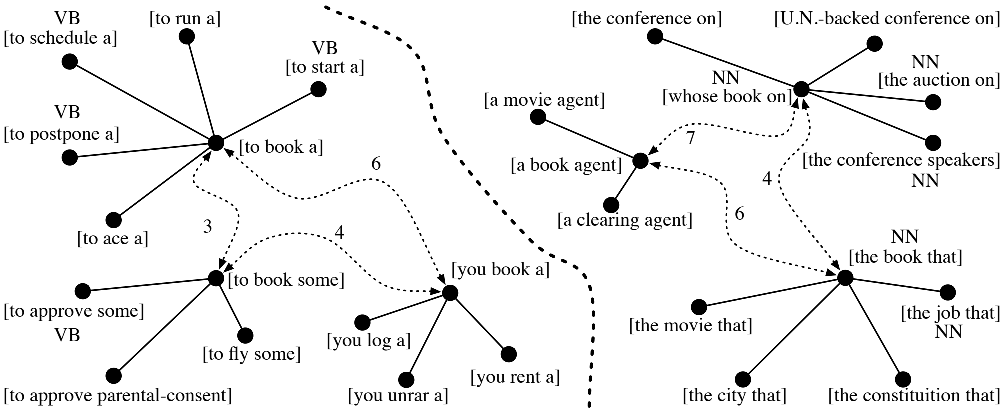


Figure 1: Vertices with center word 'book' and their local neighborhoods, as well as the shortest-path distance between them. Note that the noun (NN) and verb (VB) interpretations form two disjoint connected components.

It is remarkable that the neighborhoods are coherent, showing very similar syntactic configurations. Furthermore, different vertices that (should) have the same label are close to each other, forming connected components for each part-of-speech category (for nouns and verbs in the figure). We expect the similarity graph to provide information that cannot be expressed directly in a sequence model. In particular, it is not possible in a CRF to directly enforce the constraint that similar trigrams appearing in different sentences should have similar POS tags. This constraint however is important during (semi-supervised) learning, and is what makes our approach different and more effective than self-training.

In practice, we expect two main benefits from our graph-based approach. First, the graph allows new features to be discovered. Many words occur only in the unlabeled data and a purely supervised CRF would not be able to learn feature weights for those observations. We could use self-training to learn weights for those features, but self-training just tends to reinforce the knowledge that the supervised model already has. The similarity graph on the other hand can link events that occur only in the unlabeled data to similar events in the labeled data. Furthermore, because the graph is built over types rather than tokens, it will encourage the same interpretation to be chosen for similar trigrams occurring in different sentences. For example, the word 'unrar' will most likely not occur in the labeled training data. Seeing it in the neighborhood of words for which we know the POS tag will help us learn the correct POS tag for this otherwise unknown word (see Figure 1).

Second, the graph propagates adjustments to the weights of known features. Many words occur only a handful of times in our labeled data, resulting in poor estimates of their contributions. Even for frequently occurring events, their distribution in the target domain might be different from their distribution in the source domain. While self-training might be able to help adapt to such domain changes, its effectiveness will be limited because the model will always be inherently biased towards the source domain. In contrast, labeled vertices in the similarity graph can help disambiguate ambiguous contexts and correct (some of) the errors of the supervised model.

4 Semi-Supervised CRF

Given unlabeled data $\mathcal{D}_u$, we only have access to the prior $p(\mathbf{x})$. As the CRF is a discriminative model, the lack of label information renders the CRF weights independent of $p(\mathbf{x})$ and thus we cannot directly utilize the unlabeled data when training the CRF. Therefore, semi-supervised approaches to training discriminative models typically use the unlabeled data to construct a regularizer that is used to guide the learning process (Joachims, 1999; Lawrence and Jordan, 2005). Here we use the graph as a smoothness regularizer to train CRFs in a semi-supervised manner.

Our algorithm iterates between the following five

Algorithm 1 Semi-Supervised CRF Training

```
$\Lambda^s$ = crf-train($\mathcal{D}_l$, $\Lambda^0$)
Set $\Lambda_0^{(t)} = \Lambda^{(s)}$
while not converged do
    $\{\mathrm{p}\}$ = posterior_decode($\mathcal{D}_u$, $\Lambda_{old}$)
    $\{\mathrm{q}\}$ = token_to_type($\{\mathrm{p}\}$)
    $\{\hat{\mathrm{q}}\}$ = graph_propagate($\{\mathrm{q}\}$)
    $\mathcal{D}_u^{(1)}$ = viterbi_decode($\{\hat{\mathrm{q}}\}$, $\Lambda_{old}$)
    $\Lambda_{n+1}^{(t)}$ = crf-train($\mathcal{D}_l \cup \mathcal{D}_u^{(1)}$, $\Lambda_n^{(t)}$)
end while
Return last $\Lambda^{(t)}$
```

simple (and convex) steps: Given a set of CRF parameters, we first compute marginals over the unlabeled data (posterior_decode). The marginals over tokens are then aggregated to marginals over types (token_to_type), which are used to initialize the graph label distributions. After running label propagation (graph_propagate), the posteriors from the graph are used to smooth the state posteriors. Decoding the unlabeled data (viterbi_decode) produces a new set of automatic annotations that can be combined with the labeled data to retrain the CRF using the supervised CRF training objective (crf-train). These steps, summarized in Algorithm 1, are iterated until convergence.

4.1 Posterior Decoding

Let $\Lambda_n^{(t)}$ (t refers to target domain) represent the estimate of the CRF parameters for the target domain after the n-th iteration.[2] In this step, we use the current parameter estimates to compute the marginal probabilities

$$\mathrm{p}(y_i^{(j)}|\mathbf{x}_i;\Lambda_n^{(t)})\ 1 \le j \le |\mathbf{x}_i|, i \in \mathcal{D}_l$$

over POS tags for every word position j for i indexing over sentences in $\mathcal{D}_l \cup \mathcal{D}_u$.

4.2 Token-to-Type Mapping

Recall that our graph is defined over types while the posteriors computed above involve particular tokens. We accumulate token-based marginals to create type marginals as follows. For a sentence i and word position j in that sentence, let $T(i,j)$ be the trigram (graph node) centered at position j. Conversely, for a trigram type u, let $T^{-1}(u)$ be the set of actual occurrences (tokens) of that trigram u; that is, all pairs (i,j) where i is the index of a sentence where u occurs and j is the position of the center word of an occurrence of u in that sentence. We calculate type-level posteriors as follows:

$$\mathrm{q}_u(y) \triangleq \frac{1}{|T^{-1}(u)|} \sum_{(i,j)\in T^{-1}(u)} \mathrm{p}(y_i^{(j)}|\mathbf{x}_i;\Lambda_n^{(t)}) \quad .$$

This combination rule connects the token-centered CRF with the type-centered graph. Other ways of combining the token marginals, such as using weights derived from the entropies of marginals, might be worth investigating.

4.3 Graph Propagation

We now use our similarity graph (Section 3) to smooth the type-level marginals by minimizing the following convex objective:

$$\begin{aligned} \mathcal{C}(\mathrm{q}) &= \sum_{u\in V_l} \|\mathrm{r}_u - \mathrm{q}_u\|^2 \\ &+ \mu \sum_{u\in V, v\in\mathcal{N}(i)} w_{uv}\|\mathrm{q}_u - \mathrm{q}_v\|^2 + \nu \sum_{u\in V} \|\mathrm{q}_u - U\|^2 \\ \text{s.t.} & \sum_y \mathrm{q}_u(y) = 1\ \forall u\ \&\ \mathrm{q}_u(y) \ge 0\ \forall u, y \end{aligned} \qquad (2)$$

where $\mathrm{q} = \{\mathrm{q}_1, \mathrm{q}_2, \ldots \mathrm{q}_{|V|}\}$. The setting of the hyperparameters μ and ν will be discussed in Section 6, $\mathcal{N}(u)$ is the set of neighbors of node u, and r_u is the empirical marginal label distribution for trigram u in the labeled data. We use a squared loss to penalize neighboring nodes that have different label distributions: $\|\mathrm{q}_u - \mathrm{q}_v\|^2 = \sum_y(\mathrm{q}_u(y) - \mathrm{q}_v(y))^2$, additionally regularizing the label distributions towards the uniform distribution U over all possible labels $\mathcal{Y}$. It can be shown that the above objective is convex in q.

Our graph propagation objective can be seen as a multi-class generalization of the quadratic cost criterion (Bengio et al., 2007). The first term in the above objective requires that we respect the information in our labeled data. The second term is the graph smoothness regularizer which requires that the q_i's be smooth with respect to the graph. In other words, if w_{uv} is large, then q_u and q_v should be close in the

[2] In the first iteration, we initialize the target domain parameters to the source domain parameters: $\Lambda_0^{(t)} = \Lambda^{(s)}$.

squared-error sense. This implies that vertices u and v are likely to have similar marginals over POS tags. The last term is a regularizer and encourages all type marginals to be uniform to the extent that is allowed by the first two terms. If a unlabeled vertex does not have a path to any labeled vertex, this term ensures that the converged marginal for this vertex will be uniform over all tags, ensuring that our algorithm performs at least as well as a standard self-training based algorithm, as we will see later.

While the objective in Equation 2 admits a closed form solution, it involves inverting a matrix of order $|V|$ and thus we use instead the simple iterative update given by

$$\begin{aligned} q_u^{(m)}(y) &= \frac{\gamma_u(y)}{\kappa_u} \text{ where} \\ \gamma_u(y) &= r_u(y)\delta(u \in V_l) \\ &\quad + \sum_{v \in \mathcal{N}(u)} w_{uv} q_v^{(m-1)}(y) + \nu U(y), \\ \kappa_u &= \delta(u \in V_l) + \nu + \mu \sum_{v \in \mathcal{N}(u)} w_{uv} \end{aligned} \quad (3)$$

where m is the iteration index and δ is the indicator function that returns 1 if and only if the condition is true. The iterative procedure starts with $\mathrm{q}_u^{(0)}(y) = \mathrm{q}_u(y)$ as given in the previous section. In all our experiments we run 10 iterations of the above algorithm, and we denote the type marginals at completion by $\mathrm{q}_u^*(y)$.

4.4 Viterbi Decoding

Given the type marginals computed in the previous step, we interpolate them with the original CRF token marginals. This interpolation between type and token marginals encourages similar n-grams to have similar posteriors, while still allowing n-grams in different sentences to differ in their posteriors. For each unlabeled sentence i and word position j in it, we calculate the following interpolated tag marginal:

$$\begin{aligned} \hat{\mathrm{p}}(y_i^{(j)} = y|\mathbf{x}_i) = \alpha \mathrm{p}(y_i^{(j)} = y|\mathbf{x}_i; \Lambda_n^{(t)}) \\ + (1-\alpha)\mathrm{q}^*_{T(m,n)}(y) \end{aligned} \quad (4)$$

where α is a mixing coefficient which reflects the relative confidence between the original posteriors from the CRF and the smoothed posteriors from the graph. We discuss how we set α in Section 6.

The interpolated marginals summarize all the information obtained so far about the tag distribution at each position. However, if we were to use them on their own to select the most likely POS tag sequence, the first-order tag dependencies modeled by the CRF would be mostly ignored. This happens because the type marginals obtained from the graph after label propagation will have lost most of the sequence information. To enforce the first-order tag dependencies we therefore use Viterbi decoding over the combined interpolated marginals and the CRF transition potentials to compute the best POS tag sequence for each unlabeled sentence. We refer to these 1-best transcripts as $\mathbf{y}_i^*,\ i \in \mathcal{D}_u$.

4.5 Re-training the CRF

Now that we have successfully labeled the unlabeled target domain data, we can use it in conjunction with the source domain labeled data to re-train the CRF:

$$\begin{aligned} \Lambda_{n+1}^{(t)} = &\underset{\Lambda \in \mathbb{R}^K}{\text{argmin}} \Bigg[-\sum_{i=1}^{l} \log \mathrm{p}(\mathbf{y}_i|\mathbf{x}_i; \Lambda_n^{(t)}) \\ &- \eta \sum_{i=l+1}^{l+u} \log \mathrm{p}(\mathbf{y}_i^*|\mathbf{x}_i; \Lambda_n^{(t)}) + \gamma \|\Lambda\|^2 \Bigg] \end{aligned} \quad (5)$$

where η and γ are hyper-parameters whose setting we discuss in Section 6. Given the new CRF parameters Λ we loop back to step 1 (Section 4.1) and iterate until convergence. It is important to note that every step of our algorithm is convex, although their combination clearly is not.

5 Related Work

Our work differs from previous studies of SSL (Blitzer et al., 2006; III, 2007; Huang and Yates, 2009) for improving POS tagging in several ways. First, our algorithm can be generalized to other structured semi-supervised learning problems, although POS tagging is our motivating task and test application. Unlike III (2007), we do not require target domain labeled data. While the SCL algorithm (Blitzer et al., 2006) has been evaluated without target domain labeled data, that evaluation was to some extent transductive in that the target test data (unlabeled) was included in the unsupervised stage of SCL training that creates the structural correspondence between the two domains.

We mentioned already the algorithm of Altun et al. (2005), which is unlikely to scale up because its dual formulation requires the inversion of a matrix whose size depends on the graph size. Gupta et al. (2009) also constrain similar trigrams to have similar POS tags by forming cliques of similar trigrams and maximizing the agreement score over these cliques. Computing clique agreement potentials however is NP-hard and so they propose approximation algorithms that are still quite complex computationally. We achieve similar effects by using our simple, scalable convex graph regularization framework. Further, unlike other graph-propagation algorithms (Alexandrescu and Kirchhoff, 2009), our approach is inductive. While one might be able to make inductive extensions of transductive approaches (Sindhwani et al., 2005), these usually require extensive computational resources at test time.

6 Experiments and Results

We use the Wall Street Journal (WSJ) section of the Penn Treebank as our labeled source domain training set. We follow standard setup procedures for this task and train on sections 00-18, comprising of 38,219 POS-tagged sentences with a total of 912,344 words. To evaluate our domain-adaptation approach, we consider two different target domains: questions and biomedical data. Both target domains are relatively far from the source domain (newswire), making this a very challenging task.

The QuestionBank (Judge et al., 2006), provides an excellent corpus consisting of 4,000 questions that were manually annotated with POS tags and parse trees. We used the first half as our development set and the second half as our test set. Questions are difficult to tag with WSJ-trained taggers primarily because the word order is very different than that of the mostly declarative sentences in the training data. Additionally, the unknown word rate is more than twice as high as on the in-domain development set (7.29% vs. 3.39%). As our unlabeled data, we use a set of 10 million questions collected from anonymized Internet search queries. These queries were selected to be similar in style and length to the questions in the QuestionBank.[3] As running the CRF over 10 million sentences can be rather cumbersome and probably unnecessary, we randomly select 100,000 of these queries and treat this as $\mathcal{D}_u$. Because the graph nodes and the features used in the similarity function are based on n-grams, data sparsity can be a serious problem, and we therefore use the entire unlabeled data set for graph construction. We estimate the mutual information-based features for each trigram type over all the 10 million questions, and then construct the graph over only the set of trigram types that actually occurs in the 100,000 random subset and the WSJ training set.

For our second target domain, we use the Penn BioTreebank (PennBioIE, 2005). This corpus consists of 1,061 sentences that have been manually annotated with POS tags. We used the first 500 sentences as a development set and the remaining 561 sentences as our final test set. The high unknown word rate (23.27%) makes this corpus very difficult to tag. Furthermore, the POS tag set for this data is a super-set of the Penn Treebank's, including the two new tags HYPH (for hyphens) and AFX (for common post-modifiers of biomedical entities such as genes). These tags were introduced due to the importance of hyphenated entities in biomedical text, and are used for 1.8% of the words in the test set. Any tagger trained only on WSJ text will automatically predict wrong tags for those words. For unlabeled data we used 100,000 sentences that were chosen by searching MEDLINE for abstracts pertaining to cancer, in particular genomic variations and mutations (Blitzer et al., 2006). Since we did not have access to additional unlabeled data, we used the same set of sentences as target domain unlabeled data, $\mathcal{D}_u$. The graph here was constructed over the 100,000 unlabeled sentences and the WSJ training set. Finally, we remind the reader that we did not use label information for graph construction in either corpus.

6.1 Baselines

Our baseline supervised CRF is competitive with state-of-the-art discriminative POS taggers (Toutanova et al., 2003; Shen et al., 2007), achieving 97.17% on the WSJ development set (sections 19-21). We use a fairly standard set of features, including word identity, suffixes and prefixes and detectors

[3] In particular, we selected queries that start with an English function word that can be used to start a question (what, who, when, etc.), and have between 30 and 160 characters.

	Questions		Bio	
	Dev	Eval	Dev	Eval
Supervised CRF	84.8	83.8	86.5	86.2
Self-trained CRF	85.4	84.0	87.5	87.1
Semi-supervised CRF	**87.6**	**86.8**	**87.5**	**87.6**

Table 2: Domain adaptation experiments. POS tagging accuracies in %.

for special characters such as dashes and digits. We do not use of observation-dependent transition features. Both supervised and semi-supervised models are regularized with a squared ℓ_2-norm regularizer with weight set to 0.01.

In addition to the supervised baseline trained exclusively on the WSJ, we also consider a semi-supervised self-trained baseline ("Self-trained CRF" in Table 2). In this approach, we first train a supervised CRF on the labeled data and then do semi-supervised training without label propagation. This is different from plain self-training because it aggregates the posteriors over tokens into posteriors over types. This aggregation step allows instances of the same trigram in different sentences to share information and works better in practice than direct self-training on the output of the supervised CRF.

6.2 Domain Adaptation Results

The data set obtained concatenating the WSJ training set with the 10 million questions had about 20 million trigram types. Of those, only about 1.1 million trigram types occurred in the WSJ training set or in the 100,000 sentence sub-sample. For the biomedical domain, the graph had about 2.2 million trigrams. For all our experiments we set hyperparameters as follows: for graph propagation, $\mu = 0.5$, $\nu = 0.01$, for Viterbi decoding mixing, $\alpha = 0.6$, for CRF re-training, $\eta = 0.001$, $\gamma = 0.01$. These parameters were chosen based on development set performance. All CRF objectives were optimized using L-BFGS (Bertsekas, 2004).

Table 2 shows the results for both domains. For the question corpus, the supervised CRF performs at only 85% on the development set. While it is almost impossible to improve in-domain tagging accuracy and tagging is therefore considered a solved problem by many, these results clearly show that the problem is far from solved. Self-training improves over the baseline by about 0.6% on the development set. However the gains from self-training are more modest (0.2%) on the evaluation (test) set. Our approach is able to provide a more solid improvement of about 3% absolute over the supervised baseline and about 2% absolute over the self-trained system on the question development set. Unlike self-training, on the question evaluation set, our approach provides about 3% absolute improvement over the supervised baseline. For the biomedical data, while the performances of our approach and self-training are statistically indistinguishable on the development set, we see modest gains of about 0.5% absolute on the evaluation set. On the same data, we see that our approach provides about 1.4% absolute improvement over the supervised baseline.

7 Analysis & Conclusion

The results suggest that our proposed approach provides higher gains relative to self-training on the question data than on the biomedical corpus. We hypothesize that this caused by sparsity in the graph generated from the biomedical dataset. For the questions graph, the PMI statistics were estimated over 10 million sentences while in the case of the biomedical dataset, the same statistics were computed over just 100,000 sentences. We hypothesize that the lack of well-estimated features in the case of the biomedical dataset leads to a sparse graph.

To verify the above hypothesis, we measured the percentage of trigrams that occur in the target domain (unlabeled) data that do not have any path to a trigram in the source domain data, and the average minimum path length between a trigram in the target data and a trigram in the source data (when such a path exists). The results are shown in Table 3. For the biomedical data, close to 50% of the trigrams from the target data do not have a path to a trigram from the source data. Even when such a path exists, the average path length is about 22. On

	Questions	Bio
% of unlabeled trigrams not connected to any labeled trigrams	12.4	46.8
average path length between an unlabeled trigram and its nearest labeled trigram	9.4	22.4

Table 3: Analysis of the graphs constructed for the two datasets discussed in Section 6. Unlabeled trigrams occur in the target domain only. Labeled trigrams occur at least once in the WSJ training data.

the other hand, for the question corpus, only about 12% of the target domain trigrams are disconnected, and the average path length is about 9. These results clearly show the sparse nature of the biomedical graph. We believe that it is this sparsity that causes the graph propagation to *not* have a more noticeable effect on the final performance. It is noteworthy that making use of even such a sparse graph does not lead to any degradation in results, which we attribute to the choice of graph-propagation regularizer (Section 4.3).

We presented a simple, scalable algorithm for training structured prediction models in a semi-supervised manner. The approach is based on using as a regularizer a nearest-neighbor graph constructed over trigram types. Our results show that the approach not only scales to large datasets but also produces significantly improved tagging accuracies.

References

A. Alexandrescu and K. Kirchhoff. 2009. Graph-based learning for statistical machine translation. In *NAACL*.

Y. Altun, D. McAllester, and M. Belkin. 2005. Maximum margin semi-supervised learning for structured variables. In *Advances in Neural Information Processing Systems 18*, page 18.

M. Belkin, P. Niyogi, and V. Sindhwani. 2005. On manifold regularization. In *Proc. of the Conference on Artificial Intelligence and Statistics (AISTATS)*.

Y. Bengio, O. Delalleau, and N. L. Roux, 2007. *Semi-Supervised Learning*, chapter Label Propogation and Quadratic Criterion. MIT Press.

D Bertsekas. 2004. *Nonlinear Programming*. Athena Scientific Publishing.

J. Blitzer and J. Zhu. 2008. ACL 2008 tutorial on Semi-Supervised learning.

J. Blitzer, R. McDonald, and F. Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP '06*.

A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory*.

U. Brefeld and T. Scheffer. 2006. Semi-supervised learning for structured output variables. In *ICML06, 23rd International Conference on Machine Learning*.

O. Chapelle, B. Scholkopf, and A. Zien. 2007. *Semi-Supervised Learning*. MIT Press.

R. Collobert, F. Sinz, J. Weston, L. Bottou, and T. Joachims. 2006. Large scale transductive svms. *Journal of Machine Learning Research*.

A. Corduneanu and T. Jaakkola. 2003. On information regularization. In *Uncertainty in Artificial Intelligence*.

Y. Grandvalet and Y. Bengio. 2005. Semi-supervised learning by entropy minimization. In *CAP*.

R. Gupta, S. Sarawagi, and A. A. Diwan. 2009. Generalized collective inference with symmetric clique potentials. *CoRR*, abs/0907.0589.

G. R. Haffari and A. Sarkar. 2007. Analysis of semi-supervised learning with the Yarowsky algorithm. In *UAI*.

F. Huang and A. Yates. 2009. Distributional representations for handling sparsity in supervised sequence-labeling. In *ACL-IJCNLP '09: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1*. Association for Computational Linguistics.

H. Daume III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June. Association for Computational Linguistics.

T. Joachims. 1999. Transductive inference for text classification using support vector machines. In *Proc. of the International Conference on Machine Learning (ICML)*.

Thorsten Joachims. 2003. Transductive learning via spectral graph partitioning. In *Proc. of the International Conference on Machine Learning (ICML)*.

J. Judge, A. Cahill, and J. van Genabith. 2006. Questionbank: Creating a corpus of parse-annotated questions. In *Proceedings of the 21st International Conference on Computational Linguist ics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 497–504.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of the International Conference on Machine Learning (ICML)*.

N. D. Lawrence and M. I. Jordan. 2005. Semi-supervised learning via gaussian processes. In *NIPS*.

PennBioIE. 2005. Mining the bibliome project. In *http://bioie.ldc.upenn.edu/*.

H. J. Scudder. 1965. Probability of Error of some Adaptive Pattern-Recognition Machines. *IEEE Transactions on Information Theory*, 11.

M. Seeger. 2000. Learning with labeled and unlabeled data. Technical report, University of Edinburgh, U.K.

L. Shen, G. Satta, and A. Joshi. 2007. Guided learning for bidirectional sequence classification. In *ACL '07*.

V. Sindhwani, P. Niyogi, and M. Belkin. 2005. Beyond the point cloud: from transductive to semi-supervised learning. In *Proc. of the International Conference on Machine Learning (ICML)*.

A. Subramanya and J. A. Bilmes. 2009. Entropic graph regularization in non-parametric semi-supervised classification. In *Neural Information Processing Society (NIPS)*, Vancouver, Canada, December.

K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *HLT-NAACL '03*.

Y. Wang, G. Haffari, S. Wang, and G. Mori. 2009. A rate distortion approach for semi-supervised conditional random fields.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*.

X. Zhu, Z. Ghahramani, and J. Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proc. of the International Conference on Machine Learning (ICML)*.

X. Zhu. 2005. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.

Better Punctuation Prediction with Dynamic Conditional Random Fields

Wei Lu and **Hwee Tou Ng**
Department of Computer Science
National University of Singapore
13 Computing Drive, Singapore 117417
{luwei,nght}@comp.nus.edu.sg

Abstract

This paper focuses on the task of inserting punctuation symbols into transcribed conversational speech texts, without relying on prosodic cues. We investigate limitations associated with previous methods, and propose a novel approach based on dynamic conditional random fields. Different from previous work, our proposed approach is designed to jointly perform both sentence boundary and sentence type prediction, and punctuation prediction on speech utterances.

We performed evaluations on a transcribed conversational speech domain consisting of both English and Chinese texts. Empirical results show that our method outperforms an approach based on linear-chain conditional random fields and other previous approaches.

1 Introduction

Outputs of standard automatic speech recognition (ASR) systems typically consist of utterances where important linguistic and structural information (e.g., true case, sentence boundaries, punctuation symbols, etc) is not available. Such information is crucial in improving the readability of the transcribed speech texts, and plays an important role when further processing is required, such as in part-of-speech (POS) tagging, parsing, information extraction, and machine translation.

We focus on the punctuation prediction task in this work. Most previous punctuation prediction techniques, developed mostly by the speech processing community, exploit both lexical and prosodic cues. However, in order to fully exploit prosodic features such as pitch and pause duration, it is necessary to have access to the original raw speech waveforms. In some scenarios where further natural language processing (NLP) tasks on the transcribed speech texts become the main concern, speech prosody information may not be readily available. For example, in the recent evaluation campaign of the International Workshop on Spoken Language Translation (IWSLT) (Paul, 2009), only manually transcribed or automatically recognized speech texts are provided but the original raw speech waveforms are not available.

In this paper, we tackle the task of predicting punctuation symbols from a standard text processing perspective, where only the speech texts are available, without relying on additional prosodic features such as pitch and pause duration. Specifically, we perform the punctuation prediction task on transcribed conversational speech texts, using the IWSLT corpus (Paul, 2009) as the evaluation data.

Different from many other corpora such as broadcast news corpora, a conversational speech corpus consists of dialogs where informal and short sentences frequently appear. In addition, due to the nature of conversation, it also contains more question sentences compared to other corpora. An example English utterance randomly selected from the IWSLT corpus, along with its punctuated and cased version, are shown below:

you are quite welcome and by the way we may get other reservations so could you please call us as soon as you fix the date

You are quite welcome . And by the way , we may get other reservations , so could you please call us as soon as you fix the date ?

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 177–186,
MIT, Massachusetts, USA, 9-11 October 2010.

The rest of this paper is organized as follows. We start with surveying related work in Section 2. One class of widely-used previous techniques is then studied in detail in Section 3. Next, we investigate methods for improving existing methods in Section 4 and 5. Empirical evaluation results are presented and discussed in Section 6. We finally conclude in Section 7.

2 Related Work

Punctuation prediction has been extensively studied in the speech processing field. It is also sometimes studied together with a closely related task – sentence boundary detection.

Much previous work assumes that both lexical and prosodic cues are available for the task. Kim and Woodland (2001) performed punctuation insertion during speech recognition. Prosodic features together with language model probabilities were used within a decision tree framework. Christensen et al. (2001) focused on the broadcast news domain and investigated both finite state and multi-layer perceptron methods for the task, where prosodic and lexical information was incorporated. Huang and Zweig (2002) presented a maximum entropy-based tagging approach to punctuation insertion in spontaneous English conversational speech, where both lexical and prosodic features were exploited. Liu et al. (2005) focused on the sentence boundary detection task, by making use of conditional random fields (CRF) (Lafferty et al., 2001). Their method was shown to improve over a previous method based on hidden Markov model (HMM).

There is relatively less work that exploited lexical features only. Beeferman et al. (1998) focused on comma prediction with a trigram language model. A joint language model was learned from punctuated texts, and commas were inserted so as to maximize the joint probability score. Recent work by Gravano et al. (2009) presented a purely n-gram based approach that jointly predicted punctuation and case information of English.

Stolcke et al. (1998) presented a "hidden event language model" that treated boundary detection and punctuation insertion as an interword hidden event detection task. Their proposed method was implemented in the handy utility *hidden-ngram* as part of the SRILM toolkit (Stolcke, 2002). It was widely used in many recent spoken language translation tasks as either a preprocessing (Wang et al., 2008) or postprocessing (Kirchhoff and Yang, 2007) step. More details about this model will be given in the next section.

Recently, there are also several research efforts that try to optimize some downstream application after punctuation prediction, rather than the prediction task itself. Examples of such downstream applications include punctuation prediction for part-of-speech (POS) tagging and name tagging (Hillard et al., 2006), statistical machine translation (Matusov et al., 2006), and information extraction (Favre et al., 2008).

3 Hidden Event Language Model

Many previous research efforts consider the boundary detection and punctuation insertion task as a hidden event detection task. One such well-known approach was introduced by Stolcke et al. (1998). They adopted a HMM to describe a joint distribution over words and interword events, where the observations are the words, and the word/event pairs are encoded as hidden states. Specifically, in this task word boundaries and punctuation symbols are encoded as interword events. The training phase involves training an n-gram language model over all observed words and events with smoothing techniques. The learned n-gram probability scores are then used as the HMM state-transition scores. During testing, the posterior probability of an event at each word is computed with dynamic programming using the forward-backward algorithm. The sequence of most probable states thus forms the output which gives the punctuated sentence.

Such a HMM-based approach has several drawbacks. First, the n-gram language model is only able to capture surrounding contextual information. However, we argue that in many cases, modeling of longer range dependencies is required for punctuation insertion. For example, the method is unable to effectively capture the long range dependency between the initial phrase "*would you*" which strongly indicates a question sentence, and an ending question mark. This hurts the punctuation prediction performance for our task since we are particularly interested in conversational speech texts where question sentences appear frequently.

Thus, in practice, special techniques are usually required on top of using a hidden event language model in order to overcome long range dependencies. Examples include relocating or duplicating punctuation symbols to different positions of a sentence such that they appear closer to the indicative words (e.g., "*how much*" indicates a question sentence). One such technique was introduced by the organizers of the IWSLT evaluation campaign, who suggested duplicating the ending punctuation symbol to the beginning of each sentence before training the language model[1]. Empirically, the technique has demonstrated its effectiveness in predicting question marks in English, since most of the indicative words for English question sentences appear at the beginning of a question. However, such a technique is specially designed and may not be widely applicable in general or to languages other than English. Furthermore, a direct application of such a method may fail in the event of multiple sentences per utterance without clearly annotated sentence boundaries within an utterance.

Another drawback associated with such an approach is that the method encodes strong dependency assumptions between the punctuation symbol to be inserted and its surrounding words. Thus, it lacks the robustness to handle cases where noisy or out-of-vocabulary (OOV) words frequently appear, such as in texts automatically recognized by ASR systems. In this paper, we devise techniques based on conditional random fields to tackle the difficulties due to long range dependencies.

4 Linear-Chain Conditional Random Fields

One natural approach to relax the strong dependency assumptions encoded by the hidden event language model is to adopt an undirected graphical model, where arbitrary overlapping features can be exploited.

Conditional random fields (CRF) (Lafferty et al., 2001) have been widely used in various sequence labeling and segmentation tasks (Sha and Pereira, 2003; Tseng et al., 2005). Unlike a HMM which models the joint distribution of both the label sequence and the observation, a CRF is a discriminative model of the conditional distribution of the complete label sequence given the observation.

Specifically, a first-order linear-chain CRF which assumes first-order Markov property is defined by the following equation:

$$p_\lambda(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_t \sum_k \lambda_k f_k(\mathbf{x}, y_{t-1}, y_t, t)\right) \tag{1}$$

where $\mathbf{x}$ is the observation and $\mathbf{y}$ is the label sequence. Feature functions f_k with time step t are defined over the entire observation $\mathbf{x}$ and two adjacent hidden labels. $Z(\mathbf{x})$ is a normalization factor to ensure a well-formed probability distribution. Figure 1 gives a simplified graphical representation of the model, where only the dependencies between label and observation in the same time step are shown.

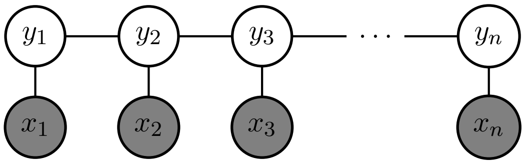


Figure 1: A simplified graphical representation for linear-chain CRF (observations are shaded)

proposed tags
NONE COMMA (,) PERIOD (.)
QMARK (?) EMARK (!)

Table 1: The set of all possible tags for linear-chain CRF

We can model the punctuation prediction task as the process of assigning a tag to each word, where the set of possible tags is given in Table 1. That is, we assume each word can be associated with an event, which tells us which punctuation symbol (possibly NONE) should be inserted after the word. The training data consists of a set of utterances where punctuation symbols are encoded as tags that are assigned to the individual words. The tag NONE means no punctuation symbol is inserted after the current word. Any other tag refers to inserting the corresponding punctuation symbol. In the testing phase, the most probable sequence of tags is

[1] http://mastarpj.nict.go.jp/IWSLT2008/downloads/case+punc_tool_using_SRILM.instructions.txt

Sentence: *no , please do not . would you save your questions for the end of my talk , when i ask for them ?*

no	*please*	*do*	*not*	*would*	*you*	...	*my*	*talk*	*when*	...	*them*
COMMA	NONE	NONE	PERIOD	NONE	NONE	...	NONE	COMMA	NONE	...	QMARK

Figure 2: An example tagging of a training sentence for the linear-chain CRF

predicted and the punctuated text can then be constructed from such an output. An example tagging of an utterance is illustrated in Figure 2.

Following (Sutton et al., 2007), we factorize a feature of conditional random fields as a product of a binary function on assignment of the set of cliques at the current time step (in this case an edge), and a feature function solely defined on the observation sequence. n-gram occurrences surrounding the current word, together with position information, are used as binary feature functions, for $n = 1, 2, 3$. All words that appear within 5 words from the current word are considered when building the features. Special start and end symbols are used beyond the utterance boundaries. For example, for the word *do* shown in Figure 2, example features include unigram features *do@0*, *please@-1*, bigram feature *would+you@[2,3]*, and trigram feature *no+please+do@[-2,0]*.

Such a linear-chain CRF model is capable of modeling dependencies between words and punctuation symbols with arbitrary overlapping features, thus avoiding the strong dependency assumptions in the hidden event language model. However, the linear-chain CRF model still exhibits several problems for the punctuation task. In particular, the dependency between the punctuation symbols and the indicative words cannot be captured adequately, if they appear too far away from each other. For example, in the sample utterance shown in Figure 2, the long range dependency between the ending question mark and the indicative words *would you* which appear very far away cannot be directly captured. The problem arises because a linear-chain CRF only learns a sequence of tags at the individual word level but is not fully aware of sentence level information, such as the start and end of a complete sentence.

Hence, it would be more reasonable to hypothesize that the punctuation symbols are annotated at the sentence level, rather than relying on a limited window of surrounding words. A model that can jointly perform sentence segmentation and sentence type prediction, together with word level punctuation prediction would be more beneficial for our task. This motivates us to build a joint model for performing such a task, to be presented in the next section.

5 Factorial Conditional Random Fields

Extensions to the linear-chain CRF model have been proposed in previous research efforts to encode long range dependencies. One such well-known extension is the semi-Markov CRF (semi-CRF) (Sarawagi and Cohen, 2005). Motivated by the hidden semi-Markov model, the semi-CRF is particularly helpful in text chunking tasks as it allows a state to persist for a certain interval of time steps. This in practice often leads to better modeling capability of chunks, since state transitions within a chunk need not precisely follow the Markov property as in the case of linear-chain CRF. However, it is not clear how such a model can benefit our task, which requires word-level labeling in addition to sentence boundary detection and sentence type prediction.

The skip-chain CRF (Sutton and McCallum, 2004), another variant of linear-chain CRF, attaches additional edges on top of a linear-chain CRF for better modeling of long range dependencies between states with similar observations. However, such a model usually requires known long range dependencies in advance and may not be readily applicable to our task where such clues are not explicit.

As we have discussed above, since we would like to jointly model both the word-level labeling task and the sentence-level annotation task (sentence boundary detection and sentence type prediction), introducing an additional layer of tags to perform both tasks together would be desirable. In this section, we propose the use of factorial CRF (F-CRF) (Sutton et al., 2007), which has previously been shown to be effective for joint labeling of multiple sequences (McCallum et al., 2003).

The F-CRF as a specific case of dynamic conditional random fields was originally motivated from dynamic Bayesian networks, where an identical structure repeats over different time steps. Analogous to the linear-chain CRF, one can think of the F-CRF as a framework that provides the capability of simultaneously labeling multiple layers of tags for a given sequence. It learns a joint conditional distribution of the tags given the observation. Formally, dynamic conditional random fields define the conditional probability of a sequence of label vectors $\mathbf{y}$ given the observation $\mathbf{x}$ as:

$$p_\lambda(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_t \sum_{c \in \mathcal{C}} \sum_k \lambda_k f_k(\mathbf{x}, y_{(c,t)}, t)\right) \tag{2}$$

where cliques are indexed at each time step, $\mathcal{C}$ is a set of clique indices, and $y_{(c,t)}$ is the set of variables in the unrolled version of a clique with index c at time t (Sutton et al., 2007). Figure 3 gives a graphical representation of a two-layer factorial CRF, where the cliques include the two within-chain edges (e.g., $z_2 - z_3$ and $y_2 - y_3$) and one between-chain edge (e.g., $z_3 - y_3$) at each time step.

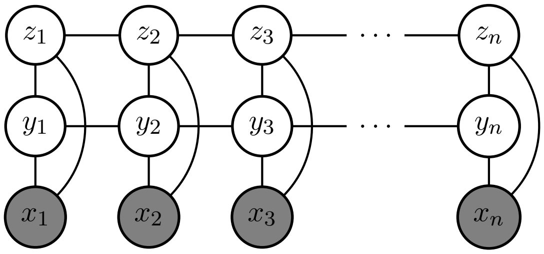


Figure 3: A two-layer factorial CRF

layer	proposed tags
word	NONE,COMMA,PERIOD, QMARK,EMARK
sentence	DEBEG,DEIN,QNBEG,QNIN, EXBEG,EXIN

Table 2: The set of all possible tags proposed for each layer

We build two layers of labels for this task, as listed in Table 2. The word layer tags are responsible for inserting a punctuation symbol (including NONE) after each word, while the sentence layer tags are used for annotating sentence boundaries and identifying the sentence type (declarative, question, or exclamatory). Tags from the word layer are the same as those of the linear-chain CRF. The sentence layer tags are designed for three types of sentences. DEBEG and DEIN indicate the start and the inner part of a declarative sentence respectively, likewise for QNBEG and QNIN (question sentences), as well as EXBEG and EXIN (exclamatory sentences). The same example utterance we looked at in the previous section is now tagged with these two layers of tags, as shown in Figure 4. Analogous feature factorization and the same n-gram feature functions used in linear-chain CRF are used in F-CRF.

When learning the sentence layer tags together with the word layer tags, the F-CRF model is capable of leveraging useful clues learned from the sentence layer about sentence type (e.g., a question sentence, annotated with QNBEG, QNIN, QNIN, ..., or a declarative sentence, annotated with DEBEG, DEIN, DEIN, ...), which can be used to guide the prediction of the punctuation symbol at each word, hence improving the performance at the word layer. For example, consider jointly labeling the utterance shown in Figure 4. Intuitively, when evidences show that the utterance consists of two sentences – a declarative sentence followed by a question sentence, the model tends to annotate the second half of the utterance with the sequence QNBEG QNIN .... This in turn helps to predict the word level tag at the end of the utterance as QMARK, given the dependencies between the two layers existing at each time step. In practice, during the learning process, the two layers of tags are jointly learned, thus providing evidences that influence each other's tagging process.

In this work, we use the GRMM package (Sutton, 2006) for building both the linear-chain CRF (L-CRF) and factorial CRF (F-CRF). The tree-based reparameterization (TRP) schedule for belief propagation (Wainwright et al., 2001) is used for approximate inference.

6 Experiments

We perform experiments on part of the corpus of the IWSLT09 evaluation campaign (Paul, 2009), where both Chinese and English conversational speech

Sentence: *no , please do not . would you save your questions for the end of my talk , when i ask for them ?*

no	*please*	*do*	*not*	*would*	*you*	...	*my*	*talk*	*when*	...	*them*
Comma	None	None	Period	None	None	...	None	Comma	None	...	QMark
DeBeg	DeIn	DeIn	DeIn	QnBeg	QnIn	...	QnIn	QnIn	QnIn	...	QnIn

Figure 4: An example tagging of a training sentence for the factorial CRF

texts are used. Two multilingual datasets are considered, the BTEC (Basic Travel Expression Corpus) dataset and the CT (Challenge Task) dataset. The former consists of tourism-related sentences, and the latter consists of human-mediated cross-lingual dialogs in travel domain. The official IWSLT09 BTEC training set consists of 19,972 Chinese-English utterance pairs, and the CT training set consists of 10,061 such pairs. We randomly split each of the two datasets into two portions, where 90% of the utterances are used for training the punctuation prediction models, and the remaining 10% for evaluating the prediction performance. For all the experiments, we use the default segmentation of Chinese as provided, and English texts are preprocessed with the Penn Treebank tokenizer[2]. We list the statistics of the two datasets after processing in Table 3. The proportions of sentence types in the two datasets are listed. The majority of the sentences are declarative sentences. However, question sentences are more frequent in the BTEC dataset compared to the CT dataset. Exclamatory sentences contribute less than 1% for all datasets and are not listed. We also count how often each utterance consists of multiple sentences. The utterances from the CT dataset are much longer (with more words per utterance), and therefore more CT utterances actually consist of multiple sentences.

	BTEC		CT	
	Cn	En	Cn	En
declarative sent.	64%	65%	77%	81%
question sent.	36%	35%	22%	19%
multi.sent./uttr.	14%	17%	29%	39%
avg.words./uttr.	8.59	9.46	10.18	14.33

Table 3: Statistics of the BTEC and CT datasets

For the methods based on the hidden event language model, we design extensive experiments due to many possible setups. Specifically, these experiments can be divided into two categories: with or without duplicating the ending punctuation symbol to the start of a sentence before training. This setting can be used to assess the impact of the proximity between the punctuation symbol and the indicative words for the prediction task. Under each category, two possible approaches are tried. The single pass approach performs prediction in one single step, where all the punctuation symbols are predicted sequentially from left to right. In the cascaded approach, we format the training sentences by replacing all sentence-ending punctuation symbols with special sentence boundary symbols first. A model for sentence boundary prediction is learned based on such training data. This step is then followed by predicting the actual punctuation symbols. Both trigram and 5-gram language models are tried for all combinations of the above settings. This gives us a total of 8 possible combinations based on the hidden event language model. When training all the language models, modified Kneser-Ney smoothing (Chen and Goodman, 1996) for n-grams is used.

To assess the performance of the punctuation prediction task, we compute precision ($prec.$), recall ($rec.$), and F1-measure (F_1), as defined by the following equations:

$$\begin{aligned} prec. &= \frac{\text{\# Correctly predicted punctuation symbols}}{\text{\# predicted punctuation symbols}} \\ rec. &= \frac{\text{\# Correctly predicted punctuation symbols}}{\text{\# expected punctuation symbols}} \\ F_1 &= \frac{2}{1/prec. + 1/rec.} \end{aligned}$$

6.1 Performance on Correctly Recognized Texts

The performance of punctuation prediction on both Chinese (Cn) and English (En) texts in the correctly recognized output of the BTEC and CT datasets are presented in Table 4 and Table 5 respectively. The

[2] http://www.cis.upenn.edu/∼treebank/tokenization.html

BTEC		No Duplication				Use Duplication				L-Crf	F-Crf
		Single Pass		Cascaded		Single Pass		Cascaded			
LM order		3	5	3	5	3	5	3	5		
Cn	*Prec.*	87.40	86.44	87.72	87.13	76.74	77.58	77.89	78.50	94.82	**94.83**
	Rec.	83.01	83.58	82.04	83.76	72.62	73.72	73.02	75.53	87.06	**87.94**
	F_1	85.15	84.99	84.79	85.41	74.63	75.60	75.37	76.99	90.78	**91.25**
En	*Prec.*	64.72	62.70	62.39	58.10	85.33	85.74	84.44	81.37	88.37	**92.76**
	Rec.	60.76	59.49	58.57	55.28	80.42	80.98	79.43	77.52	80.28	**84.73**
	F_1	62.68	61.06	60.42	56.66	82.80	83.29	81.86	79.40	84.13	**88.56**

Table 4: Punctuation prediction performance on Chinese (Cn) and English (En) texts in the correctly recognized output of the BTEC dataset. Percentage scores of precision ($Prec.$), recall ($Rec.$), and F1 measure (F_1) are reported.

CT		No Duplication				Use Duplication				L-Crf	F-Crf
		Single Pass		Cascaded		Single Pass		Cascaded			
LM order		3	5	3	5	3	5	3	5		
Cn	*Prec.*	89.14	87.83	90.97	88.04	74.63	75.42	75.37	76.87	**93.14**	92.77
	Rec.	84.71	84.16	77.78	84.08	70.69	70.84	64.62	73.60	83.45	**86.92**
	F_1	86.87	85.96	83.86	86.01	72.60	73.06	69.58	75.20	88.03	**89.75**
En	*Prec.*	73.86	73.42	67.02	65.15	75.87	77.78	74.75	74.44	83.07	**86.69**
	Rec.	68.94	68.79	62.13	61.23	70.33	72.56	69.28	69.93	76.09	**79.62**
	F_1	71.31	71.03	64.48	63.13	72.99	75.08	71.91	72.12	79.43	**83.01**

Table 5: Punctuation prediction performance on Chinese (Cn) and English (En) texts in the correctly recognized output of the CT dataset. Percentage scores of precision ($Prec.$), recall ($Rec.$), and F1 measure (F_1) are reported.

performance of the hidden event language model heavily depends on whether the duplication method is used and on the actual language under consideration. Specifically, for English, duplicating the ending punctuation symbol to the start of a sentence before training is shown to be very helpful in improving the overall prediction performance. In contrast, applying the same technique to Chinese hurts the performance.

This observed difference is reasonable and expected. An English question sentence usually starts with indicative words such as *do you* or *where* that distinguish it from a declarative sentence. Thus, duplicating the ending punctuation symbol to the start of a sentence so that it is near these indicative words helps to improve the prediction accuracy. However, Chinese presents quite different syntactic structures for question sentences. First, we found that in many cases, Chinese tends to use semantically vague auxiliary words at the end of a sentence to indicate a question. Such auxiliary words include 吗 and 呢. Thus, retaining the position of the ending punctuation symbol before training yields better performance. Another interesting finding is that, different from English, other words that indicate a question sentence in Chinese can appear at almost any position in a Chinese sentence. Examples include 哪里有... (*where* ...), ...是什么 (*what* ...), or ...多少... (*how many/much* ...). These pose difficulties for the simple hidden event language model, which only encodes simple dependencies over surrounding words by means of n-gram language modeling.

By adopting a discriminative model which exploits non-independent, overlapping features, the L-Crf model generally outperforms the hidden event language model. By introducing an additional layer of tags for performing sentence segmentation and sentence type prediction, the F-Crf model further boosts the performance over the L-Crf model. We perform statistical significance tests using bootstrap resampling (Efron et al., 1993). The improvements of F-Crf over L-Crf are statistically significant ($p < 0.01$) on Chinese and English texts in the CT

BTEC		NO DUPLICATION				USE DUPLICATION				L-CRF	F-CRF
		SINGLE PASS		CASCADED		SINGLE PASS		CASCADED			
LM ORDER		3	5	3	5	3	5	3	5		
CN	*Prec.*	85.96	84.80	86.48	85.12	66.86	68.76	68.00	68.75	92.81	**93.82**
	Rec.	81.87	82.78	83.15	82.78	63.92	66.12	65.38	66.48	85.16	**89.01**
	F_1	83.86	83.78	84.78	83.94	65.36	67.41	66.67	67.60	88.83	**91.35**
EN	*Prec.*	62.38	59.29	56.86	54.22	85.23	87.29	84.49	81.32	90.67	**93.72**
	Rec.	64.17	60.99	58.76	56.21	88.22	89.65	87.58	84.55	88.22	**92.68**
	F_1	63.27	60.13	57.79	55.20	86.70	88.45	86.00	82.90	89.43	**93.19**

Table 6: Punctuation prediction performance on Chinese (CN) and English (EN) texts in the ASR output of IWSLT08 BTEC evaluation dataset. Percentage scores of precision ($Prec.$), recall ($Rec.$), and F1 measure (F_1) are reported.

dataset, and on English texts in the BTEC dataset. The improvements of F-CRF over L-CRF on Chinese texts are smaller, probably because L-CRF is already performing quite well on Chinese. F1 measures on the CT dataset are lower than those on BTEC, mainly because the CT dataset consists of longer utterances and fewer question sentences. Overall, our proposed F-CRF model is robust and consistently works well regardless of the language and dataset it is tested on. This indicates that the approach is general and relies on minimal linguistic assumptions, and thus can be readily used on other languages and datasets.

6.2 Performance on Automatically Recognized Texts

So far we only evaluated punctuation prediction performance on transcribed texts consisting of correctly recognized words. We now present the evaluation results on texts produced by ASR systems.

For evaluation, we use the 1-best ASR outputs of spontaneous speech of the official IWSLT08 BTEC evaluation dataset, which is released as part of the IWSLT09 corpus. The dataset consists of 504 utterances in Chinese, and 498 in English. Unlike the correctly recognized texts described in Section 6.1, the ASR outputs contain substantial recognition errors (recognition accuracy is 86% for Chinese, and 80% for English (Paul, 2008)). In the dataset released by the IWSLT organizers, the correct punctuation symbols are not annotated in the ASR outputs. To conduct our experimental evaluation, we manually annotated the correct punctuation symbols on the ASR outputs.

We used all the learned models in Section 6.1, and applied them to this dataset. The evaluation results are shown in Table 6. The results show that F-CRF still gives higher performance than L-CRF and the hidden event language model, and the improvements are statistically significant ($p < 0.01$).

6.3 Performance in Translation

The evaluation process as described in Section 6.2 requires substantial manual efforts to annotate the correct punctuation symbols. In this section, we instead adopt an indirect approach to automatically evaluate the performance of punctuation prediction on ASR output texts by feeding the punctuated ASR texts to a state-of-the-art machine translation system, and evaluate the resulting translation performance. The translation performance is in turn measured by an automatic evaluation metric which correlates well with human judgments. We believe that such a task-oriented approach for evaluating the quality of punctuation prediction for ASR output texts is useful, since it tells us how well the punctuated ASR output texts from each punctuation prediction system can be used for further processing, such as in statistical machine translation.

In this paper, we use Moses (Koehn et al., 2007), a state-of-the-art phrase-based statistical machine translation toolkit, as our translation engine. We use the entire IWSLT09 BTEC training set for training the translation system. The state-of-the-art unsupervised Berkeley aligner[3] (Liang et al., 2006) is used for aligning the training bitext. We use all the default settings of Moses, except with the lexicalized reordering model enabled. This is because

[3] http://code.google.com/p/berkeleyaligner/

	No Duplication				Use Duplication				L-Crf	F-Crf
	Single Pass		Cascaded		Single Pass		Cascaded			
LM order	3	5	3	5	3	5	3	5		
Cn → En	30.77	30.71	30.98	30.64	30.16	30.26	30.33	30.42	31.27	**31.30**
En → Cn	21.21	21.00	21.16	20.76	23.03	24.04	23.61	23.34	23.44	**24.18**

Table 7: Translation performance on punctuated ASR outputs using Moses (Averaged percentage scores of Bleu)

lexicalized reordering gives better performance than simple distance-based reordering (Koehn et al., 2005). Specifically, the default lexicalized reordering model (*msd-bidirectional-fe*) is used.

For tuning the parameters of Moses, we use the official IWSLT05 evaluation set where the correct punctuation symbols are present. Evaluations are performed on the ASR outputs of the IWSLT08 BTEC evaluation dataset, with punctuation symbols inserted by each punctuation prediction method. The tuning set and evaluation set include 7 reference translations. Following a common practice in statistical machine translation, we report Bleu-4 scores (Papineni et al., 2002), which were shown to have good correlation with human judgments, with the closest reference length as the effective reference length. The minimum error rate training (MERT) (Och, 2003) procedure is used for tuning the model parameters of the translation system. Due to the unstable nature of MERT, we perform 10 runs for each translation task, with a different random initialization of parameters in each run, and report the Bleu-4 scores averaged over 10 runs.

The results are reported in Table 7. The best translation performances for both translation directions are achieved by applying F-Crf as the punctuation prediction model to the ASR texts. Such improvements are observed to be consistent over different runs. The improvement of F-Crf over L-Crf in translation quality is statistically significant ($p < 0.05$) when translating from English to Chinese. In addition, we also assess the translation performance when the manually annotated punctuation symbols as mentioned in Section 6.2 are used for translation. The averaged Bleu scores for the two translation tasks are 31.58 (Chinese to English) and 24.16 (English to Chinese) respectively, which show that our punctuation prediction method gives competitive performance for spoken language translation.

It is important to note that in this work, we only focus on optimizing the punctuation prediction performance in the form of F1-measure, without regard to the subsequent NLP tasks. How to perform punctuation prediction so as to optimize translation performance is an important research topic that is beyond the scope of this paper and needs further investigation in future work.

7 Conclusion

In this paper, we have proposed a novel approach for predicting punctuation symbols for transcribed conversational speech texts. Our proposed approach is built on top of a dynamic conditional random fields framework, which jointly performs punctuation prediction together with sentence boundary and sentence type prediction on speech utterances. Unlike most previous work, it tackles the task from a purely text processing perspective and does not rely on prosodic cues.

Experimental results have shown that our proposed approach outperforms the widely used approach based on the hidden event language model, and also outperforms a method based on linear-chain conditional random fields. Our proposed approach has been shown to be general, working well on both Chinese and English, and on both correctly recognized and automatically recognized texts. Our proposed approach also results in better translation accuracy when the punctuated automatically recognized texts are used in subsequent translation.

Acknowledgments

This research was done for CSIDM Project No. CSIDM-200804 partially funded by a grant from the National Research Foundation (NRF) administered by the Media Development Authority (MDA) of Singapore.

References

D. Beeferman, A. Berger, and J. Lafferty. 1998. CYBERPUNC: A lightweight punctuation annotation system for speech. In *Proc. of ICASSP'98*.

S.F. Chen and J. Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proc. of ACL'06*.

H. Christensen, Y. Gotoh, and S. Renals. 2001. Punctuation annotation using statistical prosody models. In *Proc. of ISCA Workshop on Prosody in Speech Recognition and Understanding*.

B. Efron, R. Tibshirani, and R.J. Tibshirani. 1993. *An introduction to the bootstrap*. Chapman & Hall/CRC.

B. Favre, R. Grishman, D. Hillard, H. Ji, D. Hakkani-Tur, and M. Ostendorf. 2008. Punctuating speech for information extraction. In *Proc. of ICASSP'08*.

A. Gravano, M. Jansche, and M. Bacchiani. 2009. Restoring punctuation and capitalization in transcribed speech. In *Proc. of ICASSP'09*.

D. Hillard, Z. Huang, H. Ji, R. Grishman, D. Hakkani-Tur, M. Harper, M. Ostendorf, and W. Wang. 2006. Impact of automatic comma prediction on POS/name tagging of speech. In *Proc. of SLT'06*.

J. Huang and G. Zweig. 2002. Maximum entropy model for punctuation annotation from speech. In *Proc. of ICSLP'02*.

J.H. Kim and P.C. Woodland. 2001. The use of prosody in a combined system for punctuation generation and speech recognition. In *Proc. of EuroSpeech'01*.

K. Kirchhoff and M. Yang. 2007. The University of Washington machine translation system for the IWSLT 2007 competition. In *Proc. of IWSLT'07*.

P. Koehn, A. Axelrod, A.B. Mayne, C. Callison-Burch, M. Osborne, and D. Talbot. 2005. Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *Proc. of IWSLT'05*.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL'07 (Demo Session)*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML'01*.

P. Liang, B. Taskar, and D. Klein. 2006. Alignment by agreement. In *Proc. of HLT/NAACL'06*.

Y. Liu, A. Stolcke, E. Shriberg, and M. Harper. 2005. Using conditional random fields for sentence boundary detection in speech. In *Proc. of ACL'05*.

E. Matusov, A. Mauser, and H. Ney. 2006. Automatic sentence segmentation and punctuation prediction for spoken language translation. In *Proc. of IWSLT'06*.

A. McCallum, K. Rohanimanesh, and C. Sutton. 2003. Dynamic conditional random fields for jointly labeling multiple sequences. In *Proc. of NIPS'03 Workshop on Syntax, Semantics and Statistics*.

F.J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of ACL'03*.

K. Papineni, S. Roukos, T. Ward, and W.J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL'02*.

M. Paul. 2008. Overview of the IWSLT 2008 evaluation campaign. In *Proc. of IWSLT'08*.

M. Paul. 2009. Overview of the IWSLT 2009 evaluation campaign. In *Proc. of IWSLT'09*.

S. Sarawagi and W.W. Cohen. 2005. Semi-Markov conditional random fields for information extraction. In *Proc. of NIPS'05*.

F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In *Proc. of HLT-NAACL'03*.

A. Stolcke, E. Shriberg, R. Bates, M. Ostendorf, D. Hakkani, M. Plauche, G. Tur, and Y. Lu. 1998. Automatic detection of sentence boundaries and disfluencies based on recognized words. In *Proc. of ICSLP'98*.

A. Stolcke. 2002. SRILM–an extensible language modeling toolkit. In *Proc. of ICSLP'02*.

C. Sutton and A. McCallum. 2004. Collective segmentation and labeling of distant entities in information extraction. In *Proc. of ICML'04 workshop on Statistical Relational Learning*.

C. Sutton, A. McCallum, and K. Rohanimanesh. 2007. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Journal of Machine Learning Research*, 8.

C. Sutton. 2006. GRMM: GRaphical Models in Mallet. `http://mallet.cs.umass.edu/grmm/`.

H. Tseng, P. Chang, G. Andrew, D. Jurafsky, and C. Manning. 2005. A conditional random field word segmenter for sighan bakeoff 2005. In *Proc. of the Fourth SIGHAN Workshop on Chinese Language Processing*.

M. Wainwright, T. Jaakkola, and A. Willsky. 2001. Tree-based reparameterization for approximate inference on loopy graphs. In *Proc. of NIPS'01*.

H. Wang, H. Wu, X. Hu, Z. Liu, J. Li, D. Ren, and Z. Niu. 2008. The TCH machine translation system for IWSLT 2008. In *Proc. of IWSLT'08*.

Joint Training and Decoding Using Virtual Nodes for Cascaded Segmentation and Tagging Tasks

Xian Qian, Qi Zhang, Yaqian Zhou, Xuanjing Huang, Lide Wu
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, P.R.China
{qianxian, qz, zhouyaqian, xjhuang, ldwu}@fudan.edu.cn

Abstract

Many sequence labeling tasks in NLP require solving a cascade of segmentation and tagging subtasks, such as Chinese POS tagging, named entity recognition, and so on. Traditional pipeline approaches usually suffer from error propagation. Joint training/decoding in the cross-product state space could cause too many parameters and high inference complexity. In this paper, we present a novel method which integrates graph structures of two subtasks into one using virtual nodes, and performs joint training and decoding in the factorized state space. Experimental evaluations on CoNLL 2000 shallow parsing data set and Fourth SIGHAN Bakeoff CTB POS tagging data set demonstrate the superiority of our method over cross-product, pipeline and candidate reranking approaches.

1 Introduction

There is a typical class of sequence labeling tasks in many natural language processing (NLP) applications, which require solving a cascade of segmentation and tagging subtasks. For example, many Asian languages such as Japanese and Chinese which do not contain explicitly marked word boundaries, word segmentation is the preliminary step for solving part-of-speech (POS) tagging problem. Sentences are firstly segmented into words, then each word is assigned with a part-of-speech tag. Both syntactic parsing and dependency parsing usually start with a textual input that is tokenized, and POS tagged.

The most commonly approach solves cascaded subtasks in a pipeline, which is very simple to implement and allows for a modular approach. While, the key disadvantage of such method is that errors propagate between stages, significantly affecting the quality of the final results. To cope with this problem, Shi and Wang (2007) proposed a reranking framework in which N-best segment candidates generated in the first stage are passed to the tagging model, and the final output is the one with the highest overall segmentation and tagging probability score. The main drawback of this method is that the interaction between tagging and segmentation is restricted by the number of candidate segmentation outputs. Razvan C. Bunescu (2008) presented an improved pipeline model in which upstream subtask outputs are regarded as hidden variables, together with their probabilities are used as probabilistic features in the downstream subtasks. One shortcoming of this method is that calculation of marginal probabilities of features may be inefficient and some approximations are required for fast computation. Another disadvantage of these two methods is that they employ separate training and the segmentation model could not take advantages of tagging information in the training procedure.

On the other hand, joint learning and decoding using cross-product of segmentation states and tagging states does not suffer from error propagation problem and achieves higher accuracy on both subtasks (Ng and Low, 2004). However, two problems arises due to the large state space, one is that the amount of parameters increases rapidly, which is apt to overfit on the training corpus, the other is that the inference by dynamic programming could be inefficient. Sutton (2004) proposed Dynamic Conditional Random Fields (DCRFs) to perform joint training/decoding of subtasks using much fewer parameters than the cross-product approach. How-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 187–195,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

ever, DCRFs do not guarantee non-violation of hard-constraints that nodes within the same segment get a single consistent tagging label. Another drawback of DCRFs is that exact inference is generally time consuming, some approximations are required to make it tractable.

Recently, perceptron based learning framework has been well studied for incorporating node level and segment level features together (Kazama and Torisawa, 2007; Zhang and Clark, 2008). The main shortcoming is that exact inference is intractable for those dynamically generated segment level features, so candidate based searching algorithm is used for approximation. On the other hand, Jiang (2008) proposed a cascaded linear model which has a two layer structure, the inside-layer model uses node level features to generate candidates with their weights as inputs of the outside layer model which captures non-local features. As pipeline models, error propagation problem exists for such method.

In this paper, we present a novel graph structure that exploits joint training and decoding in the factorized state space. Our method does not suffer from error propagation, and guards against violations of those hard-constraints imposed by segmentation subtask. The motivation is to integrate two Markov chains for segmentation and tagging subtasks into a single chain, which contains two types of nodes, then standard dynamic programming based exact inference is employed on the hybrid structure. Experiments are conducted on two different tasks, CoNLL 2000 shallow parsing and SIGHAN 2008 Chinese word segmentation and POS tagging. Evaluation results of shallow parsing task show the superiority of our proposed method over traditional joint training/decoding approach using cross-product state space, and achieves the best reported results when no additional resources at hand. For Chinese word segmentation and POS tagging task, a strong baseline pipeline model is built, experimental results show that the proposed method yields a more substantial improvement over the baseline than candidate reranking approach.

The rest of this paper is organized as follows: In Section 2, we describe our novel graph structure. In Section 3, we analyze complexity of our proposed method. Experimental results are shown in Section 4. We conclude the work in Section 5.

2 Multi-chain integration using Virtual Nodes

2.1 Conditional Random Fields

We begin with a brief review of the Conditional Random Fields(CRFs). Let $\mathbf{x} = x_1 x_2 \ldots x_l$ denote the observed sequence, where x_i is the i^{th} node in the sequence, l is sequence length, $\mathbf{y} = y_1 y_2 \ldots y_l$ is a label sequence over $\mathbf{x}$ that we wish to predict. CRFs (Lafferty et al., 2001) are undirected graphic models that use Markov network distribution to learn the conditional probability. For sequence labeling task, linear chain CRFs are very popular, in which a first order Markov assumption is made on the labels:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_i \phi(\mathbf{x}, \mathbf{y}, i)$$

,where

$$\phi(\mathbf{x}, \mathbf{y}, i) = \exp\left(\mathbf{w}^T \mathbf{f}(\mathbf{x}, y_{i-1}, y_i, i)\right)$$
$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \prod_i \phi(\mathbf{x}, \mathbf{y}, i)$$

$\mathbf{f}(\mathbf{x}, y_{i-1}, y_i, i) = [f_1(\mathbf{x}, y_{i-1}, y_i, i), \ldots, f_m(\mathbf{x}, y_{i-1}, y_i, i)]^T$, each element $f_j(\mathbf{x}, y_{i-1}, y_i, i)$ is a real valued feature function, here we simplify the notation of state feature by writing $f_j(\mathbf{x}, y_i, i) = f_j(\mathbf{x}, y_{i-1}, y_i, i)$, m is the cardinality of feature set $\{f_j\}$. $\mathbf{w} = [w_1, \ldots, w_m]^T$ is a weight vector to be learned from the training set. $Z(\mathbf{x})$ is the normalization factor over all label sequences for $\mathbf{x}$.

In the traditional joint training/decoding approach for cascaded segmentation and tagging task, each label y_i has the form s_i-t_i, which consists of segmentation label s_i and tagging label t_i. Let $\mathbf{s} = s_1 s_2 \ldots s_l$ be the segmentation label sequence over $\mathbf{x}$. There are several commonly used label sets such as BI, BIO, IOE, BIES, etc. To facilitate our discussion, in later sections we will use BIES label set, where B,I,E represents *Beginning, Inside* and *End* of a multi-node segment respectively, S denotes a single node segment. Let $\mathbf{t} = t_1 t_2 \ldots t_l$ be the tagging label sequence over $\mathbf{x}$. For example, in named entity recognition task, $t_i \in$ {PER, LOC, ORG, MISC, O} represents an entity type (person name, location name, organization name, miscellaneous entity

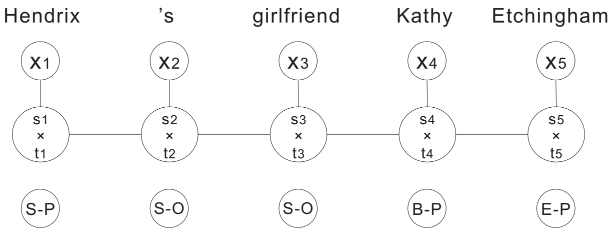


Figure 1: Graphical representation of linear chain CRFs for traditional joint learning/decoding

name and other). Graphical representation of linear chain CRFs is shown in Figure 1, where tagging label "P" is the simplification of "PER". For nodes that are labeled as other, we define s_i =S, t_i =O.

2.2 Hybrid structure for cascaded labeling tasks

Different from traditional joint approach, our method integrates two linear markov chains for segmentation and tagging subtasks into one that contains two types of nodes. Specifically, we first regard segmentation and tagging as two independent sequence labeling tasks, corresponding chain structures are built, as shown in the top and middle sub-figures of Figure 2. Then a chain of twice length of the observed sequence is built, where nodes $x_1, \dots, x_l$ on the even positions are original observed nodes, while nodes $v_1, \dots, v_l$ on the odd positions are virtual nodes that have no content information. For original nodes x_i, the state space is the tagging label set, while for virtual nodes, their states are segmentation labels. The label sequence of the hybrid chain is $\mathbf{y} = y_1 \dots y_{2l} = s_1 t_1 \dots s_l t_l$, where combination of consecutive labels $s_i t_i$ represents the full label for node x_i.

Then we let s_i be connected with s_{i-1} and s_{i+1} , so that first order Markov assumption is made on segmentation states. Similarly, t_i is connected with t_{i-1} and t_{i+1}. Then neighboring tagging and segmentation states are connected as shown in the bottom sub-figure of Figure 2. Non-violation of hard-constraints that nodes within the same segment get a single consistent tagging label is guaranteed by introducing second order transition features $f(t_{i-1}, s_i, t_i, i)$ that are true if $t_{i-1} \neq t_i$ and $s_i \in$ {I,E}. For example, $f_j(t_{i-1}, s_i, t_i, i)$ is defined as true if t_{i-1} =PER, s_i =I and t_i =LOC. In other words, it is true, if a segment is partially tagging as PER, and partially tagged as LOC. Since such features are always false in the training corpus, their corresponding weights will be very low so that inconsistent label assignments impossibly appear in decoding procedure. The hybrid graph structure can be regarded as a special case of second order Markov chain.

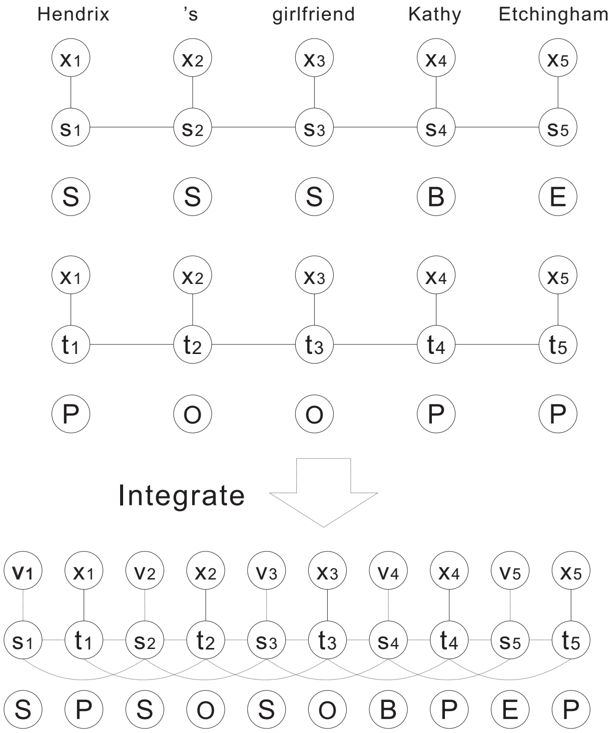


Figure 2: Multi-chain integration using Virtual Nodes

2.3 Factorized features

Compared with traditional joint model that exploits cross-product state space, our hybrid structure uses factorized states, hence could handle more flexible features. Any state feature $g(\mathbf{x}, y_i, i)$ defined in the cross-product state space can be replaced by a first order transition feature in the factorized space: $f(\mathbf{x}, s_i, t_i, i)$. As for the transition features, we use $f(s_{i-1}, t_{i-1}, s_i, i)$ and $f(t_{i-1}, s_i, t_i, i)$ instead of $g(y_{i-1}, y_i, i)$ in the conventional joint model.

Features in cross-product state space require that segmentation label and tagging label take on particular values simultaneously, however, sometimes we want to specify requirement on only segmentation or tagging label. For example, “Smith” may be an end of a person name, “Speaker: John Smith”; or a single word person name “Professor Smith will . . . ”. In such case, our observation is that “Smith” is likely a (part of) person name, we do not care about its segmentation label. So we could define state feature $f(\mathbf{x}, t_i, i) = true$, if x_i is “Smith” with tagging label t_i=PER.

Further more, we could define features like $f(\mathbf{x}, t_{i-1}, t_i, i)$, $f(\mathbf{x}, s_{i-1}, s_i, i)$, $f(\mathbf{x}, t_{i-1}, s_i, i)$, etc. The hybrid structure facilitates us to use varieties of features. In the remainder of the paper, we use notations $f(\mathbf{x}, t_{i-1}, s_i, t_i, i)$ and $f(\mathbf{x}, s_{i-1}, t_{i-1}, s_i, i)$ for simplicity.

2.4 Hybrid CRFs

A hybrid CRFs is a conditional distribution that factorizes according to the hybrid graphical model, and is defined as:

$$p(\mathbf{s}, \mathbf{t}|\mathbf{x}) \quad = \quad \frac{1}{Z(\mathbf{x})} \prod_i \phi(\mathbf{x}, \mathbf{s}, \mathbf{t}, i) \prod_i \psi(\mathbf{x}, \mathbf{s}, \mathbf{t}, i)$$

Where

$$\phi(\mathbf{x}, \mathbf{s}, \mathbf{t}, i) = \exp\left(\mathbf{w}_1^T \mathbf{f}(\mathbf{x}, s_{i-1}, t_{i-1}, s_i)\right)$$
$$\psi(\mathbf{x}, \mathbf{s}, \mathbf{t}, i) = \exp\left(\mathbf{w}_2^T \mathbf{f}(\mathbf{x}, t_{i-1}, s_i, t_i)\right)$$
$$Z(\mathbf{x}) = \sum_{\mathbf{s}, \mathbf{t}} \left(\prod_i \phi(\mathbf{x}, \mathbf{s}, \mathbf{t}, i) \prod_i \psi(\mathbf{x}, \mathbf{s}, \mathbf{t}, i) \right)$$

Where $\mathbf{w}_1$, $\mathbf{w}_2$ are weight vectors.

Luckily, unlike DCRFs, in which graph structure can be very complex, and the cross-product state space can be very large, in our cascaded labeling task, the segmentation label set is often small, so far as we known, the most complicated segmentation label set has only 6 labels (Huang and Zhao, 2007). So exact dynamic programming based algorithms can be efficiently performed.

In the training stage, we use second order forward backward algorithm to compute the marginal probabilities $p(\mathbf{x}, s_{i-1}, t_{i-1}, s_i)$ and $p(\mathbf{x}, t_{i-1}, s_i, t_i)$, and the normalization factor $Z(\mathbf{x})$. In decoding stage, we use second order Viterbi algorithm to find the best label sequence. The Viterbi decoding can be used to label a new sequence, and marginal computation is used for parameter estimation.

Table 1: Time Complexity

Method	Training	Decoding
Pipeline	$(\lvert S\rvert^2 c_s + \lvert T\rvert^2 c_t)L$	$(\lvert S\rvert^2 + \lvert T\rvert^2)U$
Cross-Product	$(\lvert S\rvert\lvert T\rvert)^2 cL$	$(\lvert S\rvert\lvert T\rvert)^2 U$
Reranking	$(\lvert S\rvert^2 c_s + \lvert T\rvert^2 c_t)L$	$(\lvert S\rvert^2 + \lvert T\rvert^2)NU$
Hybrid	$(\lvert S\rvert + \lvert T\rvert)\lvert S\rvert\lvert T\rvert cL$	$(\lvert S\rvert + \lvert T\rvert)\lvert S\rvert\lvert T\rvert U$

3 Complexity Analysis

The time complexity of the hybrid CRFs training and decoding procedures is higher than that of pipeline methods, but lower than traditional cross-product methods. Let

- $|S|$ = size of the segmentation label set.
- $|T|$ = size of the tagging label set.
- L = total number of nodes in the training data set.
- U = total number of nodes in the testing data set.
- c = number of joint training iterations.
- c_s = number of segmentation training iterations.
- c_t = number of tagging training iterations.
- N = number of candidates in candidate reranking approach.

Time requirements for pipeline, cross-product, candidate reranking and hybrid CRFs are summarized in Table 1. For Hybrid CRFs, original node x_i has features $\{f_j(t_{i-1}, s_i, t_i)\}$, accessing all label subsequences $t_{i-1}s_it_i$ takes $|S||T|^2$ time, while virtual node v_i has features $\{f_j(s_{i-1}, t_{i-1}, s_i)\}$, accessing all label subsequences $s_{i-1}t_{i-1}s_i$ takes $|S|^2|T|$ time, so the final complexity is $(|S| + |T|)|S||T|cL$.

In real applications, $|S|$ is small, $|T|$ could be very large, we assume that $|T| >> |S|$, so for each iteration, hybrid CRFs is about $|S|$ times slower than pipeline and $|S|$ times faster than cross-product

Table 2: Feature templates for shallow parsing task

Cross Product CRFs	Hybrid CRFs
$w_{i-2}y_i$, $w_{i-1}y_i$, w_iy_i	$w_{i-1}s_i$, w_is_i, $w_{i+1}s_i$
$w_{i+1}y_i$, $w_{i+2}y_i$	$w_{i-2}t_i$, $w_{i-1}t_i$, w_it_i, $w_{i+1}t_i$, $w_{i+2}t_i$
$w_{i-1}w_iy_i$, $w_iw_{i+1}y_i$	$w_{i-1}w_is_i$, $w_iw_{i+1}s_i$
	$w_{i-1}w_it_i$, $w_iw_{i+1}t_i$
$p_{i-2}y_i$, $p_{i-1}y_i$, p_iy_i	$p_{i-1}s_i$, p_is_i, $p_{i+1}s_i$
$p_{i+1}y_i$, $p_{i+2}y_i$	$p_{i-2}t_i$, $p_{i-1}t_i$, $p_{i+1}t_i$, $p_{i+2}t_i$
$p_{i-2}p_{i-1}y_i$, $p_{i-1}p_iy_i$, $p_ip_{i+1}y_i$, $p_{i+1}p_{i+2}y_i$	$p_{i-2}p_{i-1}s_i$, $p_{i-1}p_is_i$, $p_ip_{i+1}s_i$, $p_{i+1}p_{i+2}s_i$
	$p_{i-3}p_{i-2}t_i$, $p_{i-2}p_{i-1}t_i$, $p_{i-1}p_it_i$, $p_ip_{i+1}t_i$, $p_{i+1}p_{i+2}t_i$, $p_{i+2}p_{i+3}t_i$, $p_{i-1}p_{i+1}t_i$
$p_{i-2}p_{i-1}p_iy_i$, $p_{i-1}p_ip_{i+1}y_i$, $p_ip_{i+1}p_{i+2}y_i$	$p_{i-2}p_{i-1}p_is_i$, $p_{i-1}p_ip_{i+1}s_i$, $p_ip_{i+1}p_{i+2}s_i$
	$w_ip_it_i$
	$w_is_{i-1}s_i$
	$w_{i-1}t_{i-1}t_i$, $w_it_{i-1}t_i$, $p_{i-1}t_{i-1}t_i$, $p_it_{i-1}t_i$
$y_{i-1}y_i$	$s_{i-1}t_{i-1}s_i$, $t_{i-1}s_it_i$

method. When decoding, candidate reranking approach requires more time if candidate number $N > |S|$.

Though the space complexity could not be compared directly among some of these methods, hybrid CRFs require less parameters than cross-product CRFs due to the factorized state space. This is similar with factorized CRFs (FCRFs) (Sutton et al., 2004).

4 Experiments

4.1 Shallow Parsing

Our first experiment is the shallow parsing task. We use corpus from CoNLL 2000 shared task, which contains 8936 sentences for training and 2012 sentences for testing. There are 11 tagging labels: noun phrase(NP), verb phrase(VP) , . . . and other (O), the segmentation state space we used is BIES label set, since we find that it yields a little improvement over BIO set.

We use the standard evaluation metrics, which are precision P (percentage of output phrases that exactly match the reference phrases), recall R (percentage of reference phrases returned by our system), and their harmonic mean, the F1 score $F1 = \frac{2PR}{P+R}$ (which we call F score in what follows).

We compare our approach with traditional cross-product method. To find good feature templates, development data are required. Since CoNLL2000 does not provide development data set, we divide the training data into 10 folds, of which 9 folds for training and 1 fold for developing. After selecting feature templates by cross validation, we extract features and learn their weights on the whole training data set. Feature templates are summarized in Table 2, where w_i denotes the i^{th} word, p_i denotes the i^{th} POS tag.

Notice that in the second row, feature templates of the hybrid CRFs does not contain $w_{i-2}s_i$, $w_{i+2}s_i$, since we find that these two templates degrade performance in cross validation. However, $w_{i-2}t_i$, $w_{i+2}t_i$ are useful, which implies that the proper context window size for segmentation is smaller than tagging. Similarly, for hybrid CRFs, the window size of POS bigram features for segmentation is 5 (from p_{i-2} to p_{i+2}, see the eighth row in the second column); while for tagging, the size is 7 (from p_{i-3} to p_{i+3}, see the ninth row in the second column). However for cross-product method, their window sizes must be consistent.

For traditional cross-product CRFs and our hybrid CRFs, we use fixed gaussian prior $\sigma = 1.0$ for both methods, we find that this parameter does not signifi-

Table 3: Results for shallow parsing task, Hybrid CRFs significantly outperform Cross-Product CRFs (McNemar's test; $p < 0.01$)

Method	Cross-Product CRFs	Hybrid CRFs
Training Time	11.6 hours	6.3 hours
Feature Number	13 million	10 million
Iterations	118	141
F1	93.88	94.31

Table 4: Comparison with other systems on shallow parsing task

Method	F1	Additional Resources
Cross-Product CRFs	93.88	none
Hybrid CRFs	**94.31**	
SVM combination (Kudo and Matsumoto, 2001)	93.91	
Voted Perceptrons (Carreras and Marquez, 2003)	93.74	
ETL (Milidiu et al., 2008)	92.79	
(Wu et al., 2006)	94.21	Extended features such as token features, affixes
HySOL (Suzuki et al., 2007)	94.36	$17M$ words unlabeled data
ASO-semi (Ando and Zhang, 2005)	94.39	$15M$ words unlabeled data
(Zhang et al., 2002)	94.17	full parser output
(Suzuki and Isozaki, 2008)	95.15	$1G$ words unlabeled data

cantly affect the results when it varies between 1 and 10. LBFGS(Nocedal and Wright, 1999) method is employed for numerical optimization. Experimental results are shown in Table 3. Our proposed CRFs achieve a performance gain of 0.43 points in F-score over cross-product CRFs that use state space while require less training time.

For comparison, we also listed the results of previous top systems, as shown in Table 4. Our proposed method outperforms other systems when no additional resources at hand. Though recently semi-supervised learning that incorporates large mounts of unlabeled data has been shown great improvement over traditional supervised methods, such as the last row in Table 4, supervised learning is fundamental. We believe that combination of our method and semi-supervised learning will achieve further improvement.

4.2 Chinese word segmentation and POS tagging

Our second experiment is the Chinese word segmentation and POS tagging task. To facilitate comparison, we focus only on the closed test, which means that the system is trained only with a designated training corpus, any extra knowledge is not allowed, including Chinese and Arabic numbers, letters and so on. We use the Chinese Treebank (CTB) POS corpus from the Fourth International SIGHAN Bakeoff data sets (Jin and Chen, 2008). The training data consist of 23444 sentences, 642246 Chinese words, 1.05M Chinese characters and testing data consist of 2079 sentences, 59955 Chinese words, 0.1M Chinese characters.

We compare our hybrid CRFs with pipeline and candidate reranking methods (Shi and Wang, 2007) using the same evaluation metrics as shallow parsing. We do not compare with cross-product CRFs due to large amounts of parameters.

For pipeline method, we built our word segmenter based on the work of Huang and Zhao (2007), which uses 6 label representation, 7 feature templates (listed in Table 5, where c_i denotes the i^{th} Chinese character in the sentence) and CRFs for parameter learning. We compare our segmentor with other top systems using SIGHAN CTB corpus and evaluation metrics. Comparison results are shown in Table 6, our segmenter achieved 95.12 F-score, which is ranked 4th of 26 official runs. Except for the first system which uses extra unlabeled data, differences between rest systems are not significant.

Our POS tagging system is based on linear chain CRFs. Since SIGHAN dose not provide development data, we use the 10 fold cross validation described in the previous experiment to turning feature templates and Gaussian prior. Feature templates are listed in Table 5, where w_i denotes the i^{th} word in

Table 5: Feature templates for Chinese word segmentation and POS tagging task

Segmentation feature templates
(1.1) $c_{i-2}s_i$, $c_{i-1}s_i$, c_is_i, $c_{i+1}s_i$, $c_{i+2}s_i$
(1.2) $c_{i-1}c_is_i$, $c_ic_{i+1}s_i$, $c_{i-1}c_{i+1}s_i$
(1.3) $s_{i-1}s_i$
POS tagging feature templates
(2.1) $w_{i-2}t_i$, $w_{i-1}t_i$, w_it_i, $w_{i+1}t_i$, $w_{i+2}t_i$
(2.2) $w_{i-2}w_{i-1}t_i$, $w_{i-1}w_it_i$, $w_iw_{i+1}t_i$, $w_{i+1}w_{i+2}t_i$, $w_{i-1}w_{i+1}t_i$
(2.3) $c_1(w_i)t_i$, $c_2(w_i)t_i$, $c_3(w_i)t_i$, $c_{-2}(w_i)t_i$, $c_{-1}(w_i)t_i$
(2.4) $c_1(w_i)c_2(w_i)t_i$, $c_{-2}(w_i)c_{-1}(w_i)t_i$
(2.5) $l(w_i)t_i$
(2.6) $t_{i-1}t_i$
Joint segmentation and POS tagging feature templates
(3.1) $c_{i-2}s_i$, $c_{i-1}s_i$, c_is_i, $c_{i+1}s_i$, $c_{i+2}s_i$
(3.2) $c_{i-1}c_is_i$, $c_ic_{i+1}s_i$, $c_{i-1}c_{i+1}s_i$
(3.3) $c_{i-3}t_i$, $c_{i-2}t_i$, $c_{i-1}t_i$, c_it_i, $c_{i+1}t_i$, $c_{i+2}t_i$, $c_{i+3}t_i$
(3.4) $c_{i-3}c_{i-2}t_i$, $c_{i-2}c_{i-1}t_i$, $c_{i-1}c_it_i$, $c_ic_{i+1}t_i$ $c_{i+1}c_{i+2}t_i$, $c_{i+2}c_{i+3}t_i$, $c_{i-2}c_it_i$, $c_ic_{i+2}t_i$
(3.5) $c_is_it_i$
(3.6) $c_it_{i-1}t_i$
(3.7) $s_{i-1}t_{i-1}s_i$, $t_{i-1}s_it_i$

Table 6: Word segmentation results on Fourth SIGHAN Bakeoff CTB corpus

Rank	F1	Description
1/26	95.89*	official best, using extra unlabeled data (Zhao and Kit, 2008)
2/26	95.33	official second
3/26	95.17	official third
4/26	95.12	segmentor in pipeline system

Table 7: POS results on Fourth SIGHAN Bakeoff CTB corpus

Rank	Accuracy	Description
1/7	94.29	POS tagger in pipeline system
2/7	94.28	official best
3/7	94.01	official second
4/7	93.24	official third

the sentence, $c_j(w_i), j > 0$ denotes the j^{th} Chinese character of word w_i, $c_j(w_i), j < 0$ denotes the j^{th} last Chinese character, $l(w_i)$ denotes the word length of w_i. We compare our POS tagger with other top systems on Bakeoff CTB POS corpus where sentences are perfectly segmented into words, our POS tagger achieved 94.29 accuracy, which is the best of 7 official runs. Comparison results are shown in Table 7.

For reranking method, we varied candidate numbers n among $n \in \{10, 20, 50, 100\}$. For hybrid CRFs, we use the same segmentation label set as the segmentor in pipeline. Feature templates are listed in Table 5. Experimental results are shown in Figure 3. The gain of hybrid CRFs over the baseline pipeline model is 0.48 points in F-score, about 3 times higher than 100-best reranking approach which achieves 0.13 points improvement. Though larger candidate number can achieve higher performance, such improvement becomes trivial for $n > 20$.

Table 8 shows the comparison between our work and other relevant work. Notice that, such comparison is indirect due to different data sets and re-

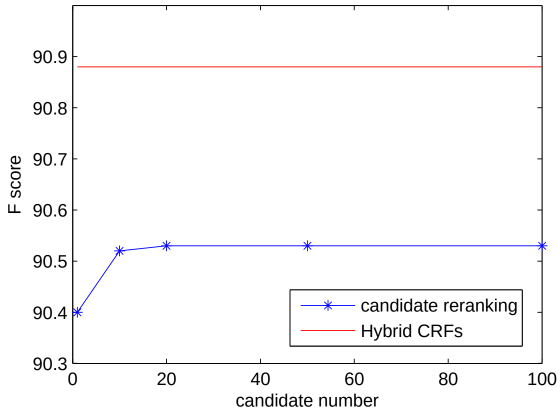


Figure 3: Results for Chinese word segmentation and POS tagging task, Hybrid CRFs significantly outperform 100-Best Reranking (McNemar's test; $p < 0.01$)

Table 8: Comparison of word segmentation and POS tagging, such comparison is indirect due to different data sets and resources.

Model	F1
Pipeline (ours)	90.40
100-Best Reranking (ours)	90.53
Hybrid CRFs (ours)	90.88
Pipeline (Shi and Wang, 2007)	91.67
20-Best Reranking (Shi and Wang, 2007)	91.86
Pipeline (Zhang and Clark, 2008)	90.33
Joint Perceptron (Zhang and Clark, 2008)	91.34
Perceptron Only (Jiang et al., 2008)	92.5
Cascaded Linear (Jiang et al., 2008)	93.4

sources. One common conclusion is that joint models generally outperform pipeline models.

5 Conclusion

We introduced a framework to integrate graph structures for segmentation and tagging subtasks into one using virtual nodes, and performs joint training and decoding in the factorized state space. Our approach does not suffer from error propagation, and guards against violations of those hard-constraints imposed by segmentation subtask. Experiments on shallow parsing and Chinese word segmentation tasks demonstrate our technique.

6 Acknowledgements

The author wishes to thank the anonymous reviewers for their helpful comments. This work was partially funded by 973 Program (2010CB327906), The National High Technology Research and Development Program of China (2009AA01A346), Shanghai Leading Academic Discipline Project (B114), Doctoral Fund of Ministry of Education of China (200802460066), National Natural Science Funds for Distinguished Young Scholar of China (61003092), and Shanghai Science and Technology Development Funds (08511500302).

References

R. Ando and T. Zhang. 2005. A high-performance semi-supervised learning method for text chunking. In *Proceedings of ACL*, pages 1–9.

Razvan C. Bunescu. 2008. Learning with probabilistic features for improved pipeline models. In *Proceedings of EMNLP*, Waikiki, Honolulu, Hawaii.

X Carreras and L Marquez. 2003. Phrase recognition by filtering and ranking with perceptrons. In *Proceedings of RANLP*.

Changning Huang and Hai Zhao. 2007. Chinese word segmentation: A decade review. *Journal of Chinese Information Processing*, 21:8–19.

Wenbin Jiang, Liang Huang, Qun Liu, and Yajuan Lu. 2008. A cascaded linear model for joint chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL*, Columbus, Ohio, USA.

Guangjin Jin and Xiao Chen. 2008. The fourth international chinese language processing bakeoff: Chinese word segmentation, named entity recognition and chinese pos tagging. In *Proceedings of Sixth SIGHAN Workshop on Chinese Language Processing*, India.

Junichi Kazama and Kentaro Torisawa. 2007. A new perceptron algorithm for sequence labeling with non-local features. In *Proceedings of EMNLP*, pages 315–324, Prague, June.

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

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

Ruy L. Milidiu, Cicero Nogueira dos Santos, and Julio C. Duarte. 2008. Phrase chunking using entropy guided transformation learning. In *Proceedings of ACL*, pages 647–655.

Hwee Tou Ng and Jin Kiat Low. 2004. Chinese partofspeech tagging: One-at-a-time or all-at-once? word-based or character-based? In *Proceedings of EMNLP*.

J. Nocedal and S. J. Wright. 1999. *Numerical Optimization*. Springer.

Yanxin Shi and Mengqiu Wang. 2007. A dual-layer crfs based joint decoding method for cascaded segmentation and labeling tasks. In *Proceedings of IJCAI*, pages 1707–1712, Hyderabad, India.

C. Sutton, K. Rohanimanesh, and A. McCallum. 2004. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *Proceedings of ICML*.

Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *Proceedings of ACL*, pages 665–673.

Jun Suzuki, Akinori Fujino, and Hideki Isozaki. 2007. Semi-supervised structured output learning based on a hybrid generative and discriminative approach. In *Proceedings of EMNLP*, Prague.

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 Intelligent Text Processing and Computational Linguistics*, pages 144–155.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and pos tagging using a single perceptron. In *Proceedings of ACL*, Columbus, Ohio, USA.

T. Zhang, F. Damerau, and D. Johnson. 2002. Text chunking based on a generalization of winnow. machine learning research. *Machine Learning Research*, 2:615–637.

Hai Zhao and Chunyu Kit. 2008. Unsupervised segmentation helps supervised learning of character tagging forword segmentation and named entity recognition. In *Proceedings of Sixth SIGHAN Workshop on Chinese Language Processing*, pages 106–111.

Crouching Dirichlet, Hidden Markov Model: Unsupervised POS Tagging with Context Local Tag Generation

Taesun Moon, Katrin Erk, and Jason Baldridge
Department of Linguistics
University of Texas at Austin
1 University Station B5100
Austin, TX 78712-0198 USA
{tsmoon,katrin.erk,jbaldrid}@mail.utexas.edu

Abstract

We define the crouching Dirichlet, hidden Markov model (CDHMM), an HMM for part-of-speech tagging which draws state prior distributions for each local document context. This simple modification of the HMM takes advantage of the dichotomy in natural language between content and function words. In contrast, a standard HMM draws all prior distributions once over all states and it is known to perform poorly in unsupervised and semi-supervised POS tagging. This modification significantly improves unsupervised POS tagging performance across several measures on five data sets for four languages. We also show that simply using different hyperparameter values for content and function word states in a standard HMM (which we call HMM+) is surprisingly effective.

1 Introduction

Hidden Markov Models (HMMs) are simple, versatile, and widely-used generative sequence models. They have been applied to part-of-speech (POS) tagging in supervised (Brants, 2000), semi-supervised (Goldwater and Griffiths, 2007; Ravi and Knight, 2009) and unsupervised (Johnson, 2007) training scenarios. Though discriminative models achieve better performance in both semi-supervised (Smith and Eisner, 2005) and supervised (Toutanova et al., 2003) learning, there has been only limited work on unsupervised discriminative sequence models (e.g., on synthetic data and protein sequences (Xu et al., 2006)), and none to POS tagging.

The tagging accuracy of purely unsupervised HMMs is far below that of supervised and semi-supervised HMMs; this is unsurprising as it is still not well understood what kind of structure is being found by an unconstrained HMM (Headden III et al., 2008). However, HMMs are fairly simple directed graphical models, and it is straightforward to extend them to define alternative generative processes. This also applies to linguistically motivated HMMs for recovering states and sequences that correspond more closely to those implicitly defined by linguists when they label sentences with parts-of-speech.

One way in which a basic HMM's structure is a poor model for POS tagging is that there is no inherent distinction between (open-class) content words and (closed-class) function words. Here, we propose two extensions to the HMM. The first, HMM+, is a very simple modification where two different hyperparameters are posited for content states and function states, respectively. The other is the *crouching Dirichlet, hidden Markov model* (CDHMM), an extended HMM that captures this dichotomy based on the statistical evidence that comes from context. Content states display greater variance across local context (e.g. sentences, paragraphs, documents), and we capture this variance by adding a component to the model for content states that is based on latent Dirichlet allocation (Blei et al., 2003). This extension is in some ways similar to the LDAHMM of Griffiths et al. (2005). Both models are composite in that two distributions do not mix with each other. Unlike the LDAHMM, the generation of content states is folded into the CDHMM process.

We compare the HMM+ and CDHMM against a basic HMM and LDAHMM on POS tagging on a more extensive and diverse set of languages than previous work in monolingual unsupervised POS tagging: four languages from three families (*Germanic*: English and German; *Romance*: Portuguese;

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 196–206,
MIT, Massachusetts, USA, 9-11 October 2010.

and *Mayan*: Uspanteko). The CDHMM easily outperforms all other models, including HMM+, across three measures (accuracy, F-score, and variation of information) for unsupervised POS tagging on most data sets. However, the HMM+ is surprisingly competitive, outperforming the basic HMM and LDAHMM, and rivaling or even passing the CDHMM on some measures and data sets.

2 Background

The Bayesian formulation for a basic HMM (Goldwater and Griffiths, 2007) is:

$$\begin{array}{rcl} \psi_t|\xi & \sim & \mathrm{Dir}(\xi) \\ \delta_t|\gamma & \sim & \mathrm{Dir}(\gamma) \\ w_i|t_i = t & \sim & \mathrm{Mult}(\psi_t) \\ t_i|t_{i-1} = t & \sim & \mathrm{Mult}(\delta_t) \end{array}$$

Dir is the conjugate Dirichlet prior to Mult (a multinomial distribution). The state transitions are generated by Mult(δ_t) whose prior δ_t is generated by Dir(γ) with a symmetric (i.e. uniform) hyperparameter γ. Emissions are generated by Mult(ψ_t) with a prior ψ_t generated by Dir(ξ) with a symmetric hyperparameter ξ. Hyperparameter values smaller than one encourage posteriors that are peaked, with smaller values increasing this concentration. It is not necessary that the hyperparameters be symmetric, but this is a common approach when one wants to be naïve about the data. This is particularly appropriate in unsupervised POS tagging with regard to novel data since there won't be *a priori* grounds for favoring certain distributions over others.

There is considerable work on extensions to HMM-based unsupervised POS tagging (see §6), but here we concentrate on the LDAHMM (Griffiths et al., 2005), which models topics and state sequences jointly. The model is a composite of a probabilistic topic model and an HMM in which a single state is allocated for words generated from the topic model. A strength of this model is that it is able to use less supervision than previous topic models since it does not require a stopword list. While the topic model component still uses the bags-of-words assumption, the joint model infers which words are more likely to carry topical content and which words are more likely to contribute to the local sequence. This model is competitive with a standard topic model, and its output is also competitive when compared with a standard HMM. However, Griffiths et al. (2005) note that the topic model component inevitably loses some finer distinctions with respect to parts-of-speech. Though many content states such as adjectives, verbs, and nouns can vary a great deal across documents, the topic state groups these words together. This leads to assignment of word tokens to clusters that are a poorer fit for POS tagging. This paper shows that a model that conflates the LDAHMM topics with content states can significantly improve POS tagging.

3 Models

We aim to model the fact that in many languages words can generally be grouped into function words and content words and that these groups often have significantly different distributions. There are few function words and they appear frequently, while there are many content words appearing infrequently. Another difference in distribution is often implied in information retrieval by the use of stopword filters and *tf-idf* values to remove or reduce the influence of words which occur frequently but have low variance (i.e. their global probability is similar to their local probability in a document).

A difference in distribution is also revealed when the parts-of-speech are known. When no smoothing parameters are added, the joint probability of a word that is not 'the' or 'a' occurring with a `DT` tag (in the Penn Treebank) is almost always zero. Similarly peaked distributions are observed for other function categories such as `MD` and `CC`. On the other hand, the joint probability of any word occurring with `NN` is much less likely to be zero and the distribution is much less likely to be peaked.

We attempt to account for these two distributional properties—that certain words have higher variance across contexts (e.g. a document) and that certain tags have more peaked emission distributions—in a sequence model. To do this, we define the *crouching Dirichlet, hidden Markov model*[1] (CDHMM). This model, like LDAHMM, captures items of high variance across contexts, but it does so without losing

[1] We call our model a "crouching Dirichlet" model since it involves a Dirichlet prior that generates distributions for certain states as if it were "crouching" on the side.

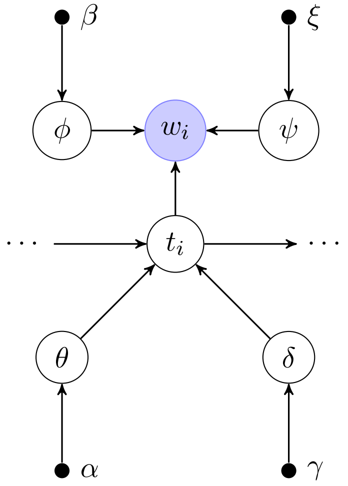


Figure 1: Graphical representation of relevant variables and dependencies at a given time step i. Observed word w_i is dependent on hidden state t_i. Edges to priors θ, ϕ, ψ may or may not be activated depending on the value of t_i. The edge to transition prior δ is always activated. Hyperparameters to priors are represented by dots. See §3.1 for details.

sequence distinctions, namely, a given word's local function via its part-of-speech. We also define the HMM+, a simple adaptation of a basic HMM which accounts for the latter property by using different priors for emissions from content and function states.

3.1 CDHMM

The CDHMM incorporates an LDA-like module to its graphical structure in order to capture words and tags which have high variance across contexts. Such tags correspond to content states. Like the LDAHMM, the model is composite in that distributions over a single random variable are composed of several different distribution functions which depend on the value of the underlying variable.

We posit the following model (see fig. 1 for a diagram of dependencies and all variables involved at a single time step). We observe a sequence of tokens $\mathbf{w}{=}(w_1, \ldots, w_N)$ that we assume is generated by an underlying state sequence $\mathbf{t}{=}(t_1, \ldots, t_N)$ over a state alphabet T with first order Markov dependencies. T is a union of disjoint content states C and function states F. In this composite model, the priors for the emission and transition for each step in the sequence depend on whether state t at step i is $t{\in}C$ or $t{\in}F$. If $t{\in}C$, the word emission is dependent on ϕ (the content word prior) and the state transition is dependent on θ (the "topic" prior) and δ (the transition prior). If $t{\in}F$, the word emission probability is dependent on ψ (the function word prior) and the state transition on δ (again, the transition prior). Therefore, if $t{\in}F$, the transition and emission structure is identical to the standard Bayesian HMM.

To elaborate, three prior distributions are defined globally for this model: (1) δ_t, the transition prior such that $p(\hat{t}|t, \delta_t) = \delta_{\hat{t}|t}$ (2) ψ_t, the function word prior such that $p(w|t, \psi_t) = \psi_{w|t}$ (3) ϕ_t, the content word prior such that $p(w|t, \phi_t) = \phi_{w|t}$. Locally for each context d (documents in our case), we define θ_d, the topic prior such that $p(t|\theta_d) = \theta_{t|d}$ for $t{\in}C$.

The generative story is as follows:

1. For each state $t{\in}T$
   (a) Draw a distribution over states $\delta_t \sim \text{Dir}(\gamma)$
   (b) If $t{\in}C$, draw a distribution over words $\phi_t \sim \text{Dir}(\beta)$
   (c) If $t{\in}F$, draw a distribution over words $\psi_t \sim \text{Dir}(\xi)$
2. For each context d
   (a) Draw a distribution $\theta_d \sim \text{Dir}(\alpha)$ over states $t{\in}C$
   (b) For each word w_i in d
      i. draw t_i from $\delta_{t_{i-1}} \circ \theta_d$
      ii. if $t_i{\in}C$, then draw w_i from ϕ_{t_i}, else draw w_i from ψ_{t_i}

For each context d, we draw a prior distribution θ_d—formally identical to the LDA topic prior—that is defined only for the states $t{\in}C$. This prior is then used to weight the draws for states at each word, from $\delta_{t_{i-1}} \circ \theta_d$, where we have defined the vector valued operation $\circ$ as follows:

$$(\delta_{t_{i-1}} \circ \theta_d)_{t_i} = \begin{cases} \frac{1}{Z}\delta_{t_i|t_{i-1}} \cdot \theta_{t_i|d} & t_i{\in}C \\ \frac{1}{Z}\delta_{t_i|t_{i-1}} & t_i{\in}F \end{cases}$$

where $(\delta_{t_{i-1}} \circ \theta_d)_{t_i}$ is the element corresponding to state t_i in the vector $\delta_{t_{i-1}} \circ \theta_d$. Z is a normalization constant such that the probability mass sums to one.

$$p(t_i|\mathbf{t}_{-i},\mathbf{w}) \propto \begin{cases} \frac{N_{w_i|t_i}+\beta}{N_{t_i}+W\beta}\ \frac{N_{t_i|d_i}+\alpha}{N_{d_i}+C\alpha}\ \frac{\left(N_{t_i|t_{i-1}}+\gamma\right)\left(N_{t_{i+1}|t_i}+\mathrm{I}[t_{i-1}=t_i=t_{i+1}]+\gamma\right)}{N_{t_i}+T\gamma+\mathrm{I}[t_i=t_{i-1}]} & t_i \in C \\ \frac{N_{w_i|t_i}+\xi}{N_{t_i}+W\xi}\ \frac{\left(N_{t_i|t_{i-1}}+\gamma\right)\left(N_{t_{i+1}|t_i}+\mathrm{I}[t_{i-1}=t_i=t_{i+1}]+\gamma\right)}{N_{t_i}+T\gamma+\mathrm{I}[t_i=t_{i-1}]} & t_i \in F \end{cases}$$

Figure 2: Conditional distribution for t_i in the CDHMM.

The important thing to note is that the draw for states at each word is proportional to a *composite* of (a) the product of the individual elements of the topic and transition priors when $t_i{\in}C$ and (b) the transition priors when $t_i{\in}F$. The draw is proportional to the product of topic and transition priors when $t_i{\in}C$ because we have made a product of experts (PoE) factorization assumption (Hinton, 2002) for tractability and to reduce the size of our model. Without such an assumption, the transition parameters would lie in a partitioned space of size $O(|C|^4)$ as opposed to $O(|T|^2)$ for the current model. Furthermore, this combination of a composite hidden state space with a product of experts assumption allows us to capture high variance for certain states.

To summarize, the CDHMM is a composite model where both the observed token and the hidden state variable are composite distributions. For the hidden state, this means that there is a "topical" element with high variance across contexts that is embedded in the state sequence for a subset of events. We embed this element through a PoE assumption where transitions into content states are modeled as a product of the transition probability and the local probability of the content state.

Inference. We use a Gibbs sampler (Gao and Johnson, 2008) to learn the parameters of this and all other models under consideration. In this inference regime, two distributions are of particular interest. One is the posterior density and the other is the conditional distribution, neither of which can be learned in closed form.

Letting $\Lambda = (\theta, \delta, \phi, \psi)$ and $h = (\alpha, \beta, \gamma, \xi)$, the posterior density is given as

$$p(\Lambda|\mathbf{w},\mathbf{t};h) \propto p(\mathbf{w},\mathbf{t}|\Lambda)p(\Lambda;h)$$

Note that $p(\mathbf{w},\mathbf{t}|\Lambda)$ is equal to

$$\prod_d^D \prod_i^{N_d} \left(\phi_{w_i|t_i}\theta_{t_i|d}\delta_{t_i|t_{i-1}}\right)^{\mathrm{I}[t_i \in C]} \left(\psi_{w_i|t_i}\delta_{t_i|t_{i-1}}\right)^{\mathrm{I}[t_i \in F]} \quad (1)$$

where I$[\cdot]$ is the indicator function, D is the number of documents in the corpus and N_d is the number of tokens in document d.

Another important measure is the conditional distribution which is conditioned on all the random variables except the hidden state variable of interest and which is derived by integrating out the priors:

$$p(t_i|\mathbf{t}_{-i},\mathbf{w};h) \propto p(t_i|\mathbf{t}_{-i};h)p(w_i|\mathbf{t},\mathbf{w}_{-i};h) \quad (2)$$

where $\mathbf{t}_{-i}$ is the joint random variable $\mathbf{t}$ without t_i and $\mathbf{w}_{-i}$ is $\mathbf{w}$ without w_i.

There are two well-known approaches to conducting Gibbs sampling for HMMs. The default method is to sample Λ based on the posterior, then sample each t_i based on the conditional distribution. Another approach is to sample directly from the conditional distribution without sampling from the posterior since the conditional distribution incorporates the posterior through integration. This is called a collapsed Gibbs sampler, which is the method employed for the models in this study.

The full conditional distribution for tag transitions for the Gibbs sampler is given in Figure 2. At each time step, we decrement all counts for the current value of t_i, sample a new value for t_i from a multinomial proportional to the conditional distribution and assign that value to t_i. β, ξ are the hyperparameters for the word emission priors of the content states and function states, respectively. γ is the hyperparameter for the state transition priors. α is the hyperparameter for the state prior given that it is in some context d. Note that we have overridden notation so that C and T here refer to the size of the alphabet. W is the size of the vocabulary. Notation such as $N_{t_i|t_{i-1}}$ refers to the counts of the events indicated by the subscript, minus the current token and tag under consideration. $N_{t_i|t_{i-1}}$ is the number of times t_i has occurred after t_{i-1} minus the tag for w_i. $N_{w_i|t_i}$ is the number of times w_i has occurred with t_i minus the current value. N_{t_i} and N_{d_i} are the counts for the given tag and document minus the current value.

In its broad outline, the CDHMM is not much more complicated than an HMM since the decomposition (eqn. 1) is nearly identical to that of an HMM with the exception that conditional probabilities for a subset of the states—the content states—are local. An inference algorithm can be derived that involves no more than adding a single term to the standard MCMC algorithm for HMMs (see Figure 2).

3.2 HMM+

The CDHMM explicitly posits two different types of states: function states and content states. Having made this distinction, there is a very simple way to capture the difference in emission distributions for function and content states within an otherwise standard HMM: posit different hyperparameters for the two types. One type has a small hyperparameter to model a sparse distribution for function words and the other has a relatively large hyperparameter to model a distribution with broader support. This extension, which we refer to as HMM+, provides an important benchmark to compare with the CDHMM to see how much is gained by its additional ability to model the fact that function words occur frequently but have low variance across contexts.

As with the CDHMM, we use Gibbs sampling to estimate the model parameters while holding the two different hyperparameters fixed. The conditional distribution for tag transitions for this model is identical to that in fig. 2 except that it does not have the second term $\frac{N_{t_i|d_i}+\alpha}{N_{d_i}+C\alpha}$ in the first case where $t_i \in C$.

We are not aware of a published instance of such an extension to the HMM—which our results show to be surprisingly effective. Goldwater and Griffiths (2007) posits different hyperparameters for individual states, but not for different groups of states.

corpus	tokens	docs	avg.	tags
WSJ	974254	1801	541	43
Brown	797328	343	2325	80
Tiger	447079	1090	410	58
Floresta	197422	1956	101	19
Uspanteko	70125	29	2418	83

Table 2: Number of tokens, documents, average tokens per document and total tag types for each corpus.

4 Data and Experiments

Data. We use five datasets from four languages (English, German, Portuguese, Uspanteko) for evaluating POS tagging performance.

- *English*: the Brown corpus (Francis et al., 1982) and the Wall Street Journal portion of the Penn Treebank (Marcus et al., 1994).
- *German*: the Tiger corpus (Brants et al., 2002).
- *Portuguese*: the full Bosque subset of the Floresta corpus (Afonso et al., 2002).
- *Uspanteko* (an endangered Mayan language of Guatemala): morpheme-segmented and POS-tagged texts collected and annotated by the OKMA language documentation project (Pixabaj et al., 2007); we use the cleaned-up version described in Palmer et al. (2009).

Table 2 provides the statistics for these corpora.

We lowercase all words, do not remove any punctuation or *hapax legomena*, and we do not replace numerals with a single identifier. Due to the nature of the models, document boundaries are retained.

Evaluation We report values for three evaluation metrics on all five corpora, using their full tagsets.

- *Accuracy*: We use a greedy search algorithm to map each unsupervised tag to a gold label such that accuracy is maximized. We evaluate on a **1-to-1** mapping between unsupervised tags and gold labels, as well as many-to-1 (**M-to-1**), corresponding to the evaluation mappings used in Johnson (2007). The 1-to-1 mapping provides a stricter evaluation. The many-to-one mapping, on the other hand, may be more adequate as unsupervised tags tend to be more fine-grained than

	Model	Accuracy		Pairwise P/R Scores			VI
		1-to-1	M-to-1	P	R	F	
WSJ (50)	HMM	0.34 (0.01)	0.49 (0.03)	**0.51** (0.03)	0.19 (0.01)	0.28 (0.01)	3.72 (0.08)
	LDAHMM	0.30 (0.04)	0.45 (0.04)	0.25 (0.07)	0.27 (0.03)	0.26 (0.04)	3.64 (0.14)
	HMM+	***0.42*** (0.04)	0.46 (0.05)	0.24 (0.03)	**0.49** (0.03)	0.32 (0.03)	***2.65*** (0.15)
	CDHMM	***0.44*** (0.01)	**0.58** (0.02)	0.31 (0.01)	0.43 (0.03)	**0.36** (0.02)	***2.73*** (0.08)
Brown (50)	HMM	0.32 (0.01)	0.50 (0.02)	**0.60** (0.02)	0.18 (0.00)	0.28 (0.01)	3.82 (0.05)
	LDAHMM	0.28 (0.06)	0.41 (0.08)	0.25 (0.10)	0.28 (0.05)	0.25 (0.05)	3.71 (0.21)
	HMM+	0.43 (0.06)	0.48 (0.07)	0.29 (0.05)	0.50 (0.04)	***0.37*** (0.05)	2.63 (0.19)
	CDHMM	**0.48** (0.02)	**0.62** (0.02)	0.32 (0.03)	**0.54** (0.04)	***0.40*** (0.03)	**2.48** (0.06)
Tiger (50)	HMM	0.29 (0.02)	0.49 (0.02)	**0.49** (0.04)	0.14 (0.01)	0.22 (0.02)	3.91 (0.06)
	LDAHMM	0.31 (0.04)	0.50 (0.04)	0.26 (0.07)	0.24 (0.02)	0.25 (0.04)	3.51 (0.11)
	HMM+	0.41 (0.08)	0.44 (0.05)	0.25 (0.05)	***0.58*** (0.10)	0.35 (0.06)	***2.70*** (0.25)
	CDHMM	**0.47** (0.01)	**0.61** (0.02)	0.45 (0.01)	***0.58*** (0.03)	**0.50** (0.02)	***2.72*** (0.04)
Usp. (50)	HMM	0.36 (0.01)	***0.49*** (0.02)	**0.39** (0.01)	0.18 (0.00)	**0.25** (0.00)	3.63 (0.04)
	LDAHMM	0.35 (0.02)	0.47 (0.02)	0.26 (0.04)	0.23 (0.03)	0.24 (0.02)	3.52 (0.09)
	HMM+	0.32 (0.02)	0.35 (0.03)	0.12 (0.02)	**0.52** (0.05)	0.20 (0.02)	3.13 (0.06)
	CDHMM	**0.39** (0.02)	***0.50*** (0.02)	0.16 (0.02)	0.39 (0.03)	0.23 (0.02)	**3.00** (0.06)
Flor. (50)	HMM	0.30 (0.01)	0.58 (0.03)	**0.62** (0.05)	0.18 (0.01)	0.28 (0.01)	3.51 (0.06)
	LDAHMM	***0.36*** (0.06)	0.59 (0.04)	0.55 (0.10)	0.29 (0.07)	***0.38*** (0.08)	3.22 (0.15)
	HMM+	0.35 (0.04)	0.52 (0.02)	0.28 (0.04)	**0.43** (0.06)	***0.34*** (0.04)	**2.58** (0.07)
	CDHMM	***0.36*** (0.01)	**0.64** (0.02)	0.37 (0.02)	0.27 (0.01)	0.31 (0.01)	2.73 (0.05)

Table 1: Evaluation on WSJ, Brown, Tiger, Floresta and Uspanteko for models with 50 states. For VI, lower is better

gold part-of-speech tags. In particular, they tend to form semantically coherent sub-classes of gold parts of speech.

- *Pairwise Precision and Recall*: Viewing tagging as a clustering task over tokens, we evaluate pairwise precision (P) and recall (R) between the model tag sequence (M) and gold tag sequence (G) by counting the true positives (tp), false positives (fp) and false negatives (fn) between the two and setting $P = tp/(tp + fp)$ and $R = tp/(tp + fn)$. tp is the number of token pairs that share a tag in M as well as in G, fp is the number token pairs that share the same tag in M but have different tags in G, and fn is the number token pairs assigned a different tag in M but the same in G (Meila, 2007). We also provide the f-score which is the harmonic mean of P and R.
- *Variation of Information (VI)*: The variation of information is an information theoretic metric that measures the amount of information lost and gained in going from tag sequence M to G (Meila, 2007). It is defined as $VI(M, G) = H(M) + H(G) - 2I(M, G)$ where H denotes entropy and I mutual information. Goldwater and Griffiths (2007) noted that this measure can point out models that have more consistent errors in the form of lower VI, even when accuracy figures are the same.

We also report learning curves on **M-to-1** with geometrically increasing training set sizes of 8, 16, 32, 64, 128, 256, 512, 1024, and all documents, or as many as possible given the corpus.

5 Experiments

In this section we discuss our parameter settings and experimental results.

5.1 Models and Parameters

We compare four different models:

- HMM: a standard HMM
- HMM+: an HMM in which the hyperparameters for the word emissions are asymmetric, such that content states have different word emission priors compared to function states.
- LDAHMM: an HMM with a distinguished state that generates words from a topic model (Griffiths et al., 2005)

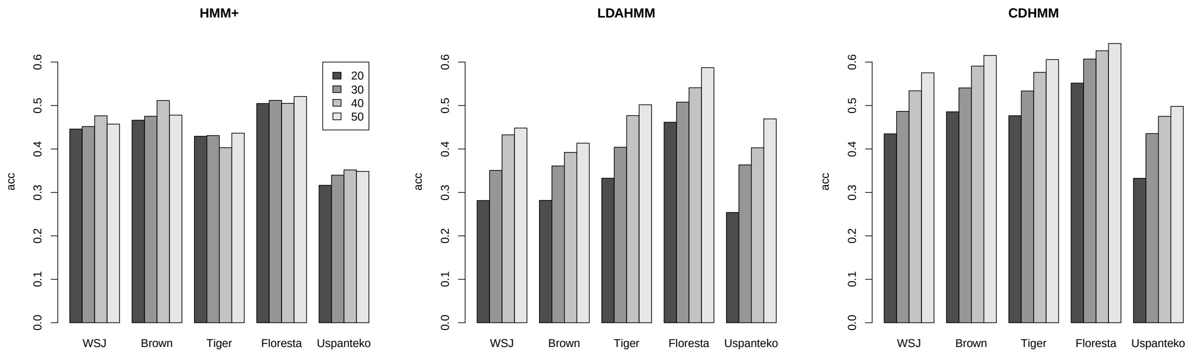


Figure 3: Averaged many-to-one accuracy on the full tagset for the models HMM+, LDAHMM, CDHMM when the number of states is set at 20, 30, 40 and 50 states.

- CDHMM: our HMM with context-based emissions, where the context used is the document

We implemented all of these models, ensuring performance differences are due to the models themselves rather than implementation details.

For all models, the transition hyperparameters γ are set to 0.1. For the LDAHMM and HMM all emission hyperparameters are set to 0.0001. These figures are the MCMC settings that provided the best results in Johnson (2007). For the models that distinguish content and function states (HMM+, CDHMM), we fixed the number of content states at 5 and set the function state emission hyperparameters $\xi = 0.0001$ and the content state emission hyperparameters $\beta = 0.1$. For the models with an LDA or LDA-like component (LDAHMM, CDHMM), we set the topic or content-state hyperparameter $\alpha = 1$.

For decoding, we use maximum posterior decoding to obtain a single sample after the required burn-in, as has been done in other unsupervised HMM experiments. We use this sample for evaluation.

5.2 Results

Results for all models on the full tagset are provided in table 1.[2] Each number is the mean accuracy of ten randomly initialized samples after a single chain burn-in of 1000 iterations. The model with a statistically significant ($p < 0.05$) best score for each measure and data set is given in plain bold. In cases where the differences for the best models are not significantly different from each other, but are significantly better from the others, the top model scores are given in bold italic.

CDHMM is extremely strong on the accuracy metric: it wins or ties for all datasets for both 1-to-1 and M-to-1 measures. For pairwise f-score, it obtains the best score for two datasets (WSJ and Tiger), and ties with HMM+ on Brown (we return to Uspanteko and Floresta below in an experiment that varies the number of states). For VI, HMM+ and CDHMM both easily outperform the other models, with CDHMM winning Brown and Uspanteko and HMM+ winning Floresta.

In the case of Uspanteko, the absolute difference in mean performance between models is smaller overall but still significant. This is due to the reduced variance between samples for all models. This is striking because the non-CDHMM models have much higher standard deviation on other corpora but have sharply reduced standard deviation only for Uspanteko. The most likely explanation is that the Uspanteko corpus is much smaller than the other corpora.[3] Nonetheless, CDHMM comes out strongest on most measures.

A simple baseline for accuracy is to choose the most frequent tag for all tokens; this gives accuracies of 0.14 (WSJ), 0.14 (Brown), 0.21 (Tiger), 0.20

[2] Similar results are obtained with reduced tagsets, as is commonly done in other work on unsupervised POS-tagging.

[3] which is interesting in itself since the weak law of large numbers implies that sample standard deviation decreases with sample size, which in our case is the number of tokens rather than the 10 samples under discussion

	Model	Accuracy		P/R Scores			VI
		1-to-1	M-to-1	P	R	F	
Usp. (100)	HMM	0.36 (0.01)	0.58 (0.01)	**0.56** (0.02)	0.16 (0.00)	0.25 (0.01)	3.53 (0.04)
	LDAHMM	0.35 (0.01)	0.58 (0.02)	0.45 (0.04)	0.17 (0.01)	0.24 (0.01)	3.46 (0.06)
	HMM+	0.35 (0.02)	0.41 (0.02)	0.18 (0.01)	**0.36** (0.03)	0.24 (0.01)	3.25 (0.08)
	CDHMM	**0.40** (0.01)	**0.59** (0.01)	0.25 (0.02)	0.27 (0.02)	**0.26** (0.01)	**3.05** (0.03)
Flor. (20)	HMM	0.31 (0.02)	0.48 (0.03)	**0.40** (0.03)	0.21 (0.01)	0.28 (0.02)	3.54 (0.10)
	LDAHMM	0.35 (0.06)	0.46 (0.06)	0.27 (0.07)	0.45 (0.08)	0.33 (0.05)	3.10 (0.10)
	HMM+	0.37 (0.04)	0.50 (0.03)	0.30 (0.02)	0.45 (0.06)	0.36 (0.03)	2.62 (0.06)
	CDHMM	**0.44** (0.02)	**0.55** (0.02)	0.30 (0.01)	**0.53** (0.03)	**0.39** (0.02)	**2.39** (0.07)

Table 3: Evaluation for Uspanteko and Floresta. Experiments in this table use state sizes that correspond more closely to the size of the tag sets in the respective corpora.

(Floresta), and 0.11 (Uspanteko). Clearly, all of the models easily outperform this baseline.

Number of states. Figure 3 shows the change in accuracy for the different models for different corpora when the overall number of states is varied between 20 and 50. The figure shows results for **M-to-1**. All models with the exception of HMM+ show improvements as the number of states is increased. This brings up the valid concern (Clark, 2003; Johnson, 2007) that a model could posit a very large number of states and obtain high M-to-1 scores. However, it is neither the case here nor in any of the studies we cite. Furthermore, as is strongly suggested with HMM+, it does not seem as if all models will benefit from assuming a large number of states.

Looking at the results by number of states on VI and f-score for CDHMM(Figure 5), it is clear that Floresta displays the reverse pattern of all other data sets where performance monotonically deteriorates as state sizes are increased. Though the exact reason is unknown, we believe it is partially due to the fact that Floresta has 19 tags. We therefore wondered whether positing a state size that more closely approximated the size of the gold tag set performs better. Since the discrepancy is greatest for Uspanteko and Floresta, we present tabulated results for experiments with state settings of 100 and 20 states respectively (table 3). With the exception of VI (where lower is better) for Uspanteko, the scores generally improve when the model state size is closer to the gold size. **M-to-1** goes down for Floresta when 20 states are posited, but this is to be expected since this score is defined, to a certain extent, to do better with larger models.

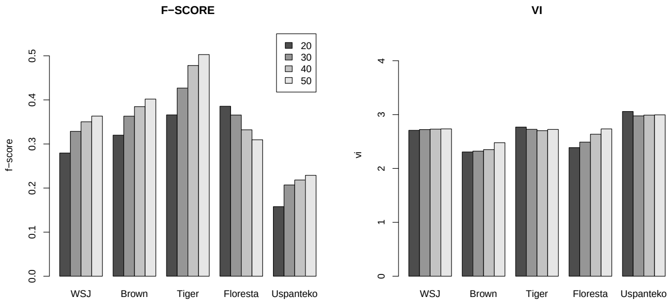


Figure 5: f-score and VI for CDHMM by number of states

Variance. As we average performance figures over ten runs for each model, it is also instructive to consider standard deviation across runs. Standard deviation is lowest for the CDHMM models and the vanilla HMM. Standard deviation is high for HMM+ and LDAHMM. This is not surprising for LDAHMM, since it has fifty topic parameters in addition to the number of states posited, and random initial conditions would have greater effect on the outcome than for the other models. It is unexpected, however, that HMM+ has high variance over different chains. The model shares the large content emission hyperparameter $\beta = 0.1$ with CDHMM. At this point, it can only be assumed that the additional LDA component acts as a regularization factor for CDHMM and reduced the volatility in having a large emission hyperparameter.

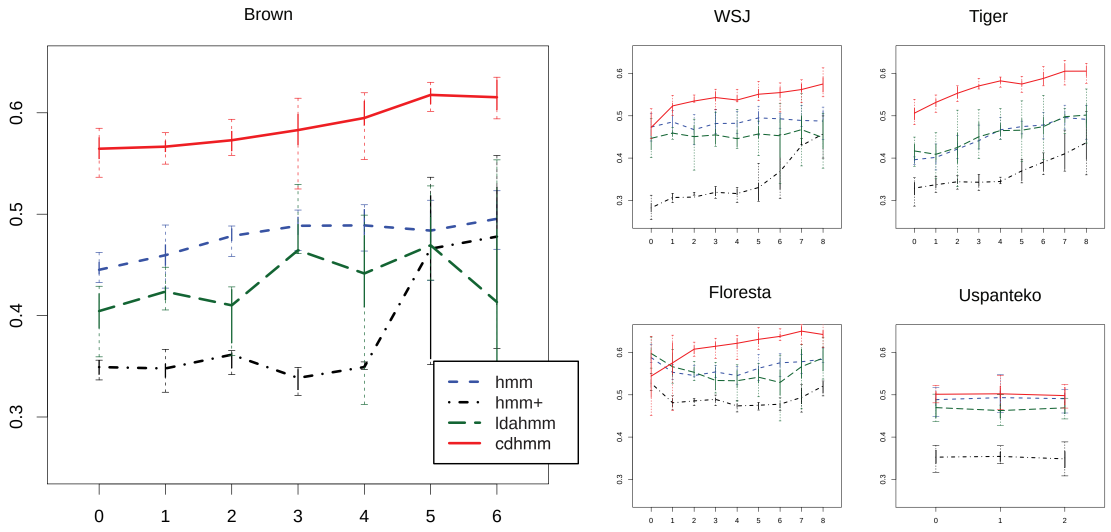


Figure 4: Learning curves on M-to-1 evaluation. The staples at each point represent two standard deviations.

Learning curves We present learning curves on different sizes of subcorpora in Figure 4. The graphs are box plots of the full M-1 accuracy figures on 10 randomly initialized training runs for seven subcorpora in Brown, nine in WSJ, Tiger, Floresta and three in Uspanteko.

Comparing the graphs, the performance of HMM+ shows the strongest improvement for English and German data as the amount of training data increases. Also, it is evident that CDHMM posts consistent performance gains across data sets as it trains on more data. This stands in opposition to HMM and LDAHMM which do not seem able to take advantage of more information for WSJ and Floresta. This suggests that performance for CDHMM and HMM+ could improve if the training corpora were augmented with out-of-corpus raw data. One exception to the consistent improvement over increased data is the performance of the models on Uspanteko, which uniformly flatline. One reason might be that the tags are labeled over segmented morphemes instead of words like the other corpora. Another could be that Uspanteko has a relatively large number of tags in a very small corpus.

6 Related work

Unsupervised POS tagging is an active area of research. Most recent work has involved HMMs. Given that an unconstrained HMM is not well understood in POS tagging, much work has been done on examining the mechanism and the properties of the HMM as applied to natural language data (Johnson, 2007; Gao and Johnson, 2008; Headden III et al., 2008). Conversely, there has also been work focused on improving the HMM as an inference procedure that looked at POS tagging as an example (Graca et al., 2009; Liang and Klein, 2009). Nonparametric HMMs for unsupervised POS tag induction (Snyder et al., 2008; Van Gael et al., 2009) have seen particular activity due to the fact that model size assumptions are unnecessary and it lets the data "speak for itself."

There is also work on alternative unsupervised models that are not HMMs (Schütze, 1993; Abend et al., 2010; Reichart et al., 2010b) as well as research on improving evaluation of unsupervised taggers (Frank et al., 2009; Reichart et al., 2010a).

Though they did not concentrate on unsupervised methods, Haghighi and Klein (2006) conducted an unsupervised experiment that utilized certain token features (e.g. character suffixes of 3 or less, has initial capital, etc.; the features themselves are from Smith and Eisner (2005)) to learn parameters in an undirected graphical model which was the equivalent of an HMM in directed models. It was also the first study to posit the one-to-one evaluation criterion which has been repeated extensively since (Johnson, 2007; Headden III et al., 2008; Graca et al., 2009).

Finkel et al. (2007) is an interesting variant of unsupervised POS tagging where a parse tree is assumed and POS tags are induced from this structure non-parametrically. It is the converse of unsupervised parsing which assumes access to a tagged corpus and induces a parsing model.

Other models more directly influenced or closely parallel our work. Griffiths et al. (2005) is the work that inspired the current approach where a set of states is designated to capture variance across contexts. The primary goal of that model was to induce a topic model given data that had not been filtered of noise in the form of function words. As such, distinguishing between topic states such that they model different syntactic states was not attempted, and we have seen in sec. 3 that such an extension is not entirely straightforward.[4] Boyd-Graber and Blei (2009) has some parallels to our model in that a hidden variable over topics is distributed according to a normalized product between a context prior and a syntactic prior. However, it assumes a much greater amount of information than we do in that a parse tree as well as (possibly) POS tags are taken as observed. The model has a very different goal from ours as well, which is to infer a syntactically informed topic model. Teichert and Daumé III (2010) is another study with close similarities to our own. This study models distinctions between closed class words and open class words within a modified HMM. It is unclear from their formulation how the distinction between open class and closed class words is learned.

There is also extensive literature on learning sequence structure from *unlabeled* text (Smith and Eisner, 2005; Goldberg et al., 2008; Ravi and Knight, 2009) which assume access to a tag dictionary. Goldwater and Griffiths (2007) deserves mention for examining a semi-supervised model that sampled emission hyperparameters for each state rather than a single symmetric hyperparameter. They showed that this outperformed a symmetric model. An interesting heuristic model is Zhao and Marcus (2009) that uses a seed set of closed class words to classify open class words.

7 Conclusion

We have shown that a hidden Markov model that allocates a subset of the states to have distributions conditioned on localized domains can significantly improve performance in unsupervised part-of-speech tagging. We have also demonstrated that significant performance gains are possible simply by setting a different emission hyperparameter for a subgroup of the states. It is encouraging that these results hold for both models not just on the WSJ but across a diverse set of languages and measures.

We believe our proposed extensions to the HMM are a significant contribution to the general HMM and unsupervised POS tagging literature in that both can be implemented with minimum modification of existing MCMC inferred HMMs, have (nearly) equivalent run times, produce output that is easy to interpret since they are based on a generative framework, and bring about considerable performance improvements at the same time.

Acknowledgments

The authors would like to thank Elias Ponvert and the anonymous reviewers. This work was supported by a grant from the Morris Memorial Trust Fund of the New York Community Trust.

References

O. Abend, R. Reichart, and A. Rappoport. 2010. Improved unsupervised POS induction through prototype discovery. In *Proceedings of ACL*, pages 1298–1307.

S. Afonso, E. Bick, R. Haber, and D. Santos. 2002. Floresta sintá(c)tica": a treebank for Portuguese. In *Proceedings of LREC*, pages 1698–1703.

D. M. Blei, A. Y. Ng, and M. I. Jordan. 2003. Latent Dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022.

J. L. Boyd-Graber and D. Blei. 2009. Syntactic topic models. In *Proceedings of NIPS*, pages 185–192.

[4] We tested a variant of LDAHMM in which more than one state can generate topics. It did not achieve good results.

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*.

T. Brants. 2000. TnT: a statistical part-of-speech tagger. In *Proceedings of conference on Applied natural language processing*, pages 224–231.

A. Clark. 2003. Combining distributional and morphological information for part of speech induction. In *Proceedings of EACL*, pages 59–66.

J. R. Finkel, T. Grenager, and C. D. Manning. 2007. The infinite tree. In *Proceedings of ACL*, pages 272–279.

W.N. Francis, H. Kučera, and A.W. Mackie. 1982. *Frequency analysis of English usage: Lexicon and grammar*. Houghton Mifflin Harcourt.

S. Frank, S. Goldwater, and F. Keller. 2009. Evaluating models of syntactic category acquisition without using a gold standard. In *Proceedings of CogSci*.

J. Gao and M. Johnson. 2008. A comparison of Bayesian estimators for unsupervised Hidden Markov Model POS taggers. In *Proceedings of EMNLP*, pages 344–352.

Y. Goldberg, M. Adler, and M. Elhadad. 2008. EM can find pretty good HMM POS-taggers (when given a good start). In *Proceedings of ACL*, pages 746–754.

S. Goldwater and T. L. Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of ACL*, pages 744–751.

J. Graca, K. Ganchev, B. Taskar, and F. Pereira. 2009. Posterior vs parameter sparsity in latent variable models. In *Proceedings of NIPS*, pages 664–672.

T. L. Griffiths, M. Steyvers, D. M. Blei, and J. M. Tenenbaum. 2005. Integrating topics and syntax. In *Proceedings of NIPS*, pages 537–544.

A. Haghighi and D. Klein. 2006. Prototype-driven learning for sequence models. In *Proceedings of HLT/NAACL*, pages 320–327.

W. P. Headden III, D. McClosky, and E. Charniak. 2008. Evaluating unsupervised part-of-speech tagging for grammar induction. In *Proceedings of COLING*, pages 329–336.

G.E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

M. Johnson. 2007. Why doesn’t EM find good HMM POS-taggers. In *Proceedings of EMNLP-CoNLL*, pages 296–305.

P. Liang and D. Klein. 2009. Online EM for unsupervised models. In *Proceedings of HLT/NAACL*, pages 611–619.

M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Comp. ling.*, 19(2):313–330.

M. Meila. 2007. Comparing clusterings—an information based distance. *Journal of Multivariate Analysis*, 98(5):873–895.

A. Palmer, T. Moon, and J. Baldridge. 2009. Evaluating automation strategies in language documentation. In *Proceedings of the NAACL-HLT 2009 Workshop on Active Learning for Natural Language Processing*, pages 36–44.

T. C. Pixabaj, M. A. Vicente Méndez, M. Vicente Méndez, and O. A. Damián. 2007. Text Collections in Four Mayan Languages. Archived in The Archive of the Indigenous Languages of Latin America.

S. Ravi and K. Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *Proceedings of ACL and AFNLP*, pages 504–512.

R. Reichart, O. Abend, and A. Rappoport. 2010a. Type level clustering evaluation: New measures and a POS induction case study. In *Proceedings of CoNLL*, pages 77–87.

R. Reichart, R. Fattal, and A. Rappoport. 2010b. Improved unsupervised POS induction using intrinsic clustering quality and a Zipfian constraint. In *Proceedings of CoNLL*, pages 57–66.

H. Schütze. 1993. Part-of-speech induction from scratch. In *Proceedings of ACL*, pages 251–258.

N.A. Smith and J. Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of ACL*, pages 354–362.

B. Snyder, T. Naseem, J. Eisenstein, and R. Barzilay. 2008. Unsupervised multilingual learning for POS tagging. In *Proceedings of EMNLP*, pages 1041–1050.

A.R. Teichert and H. Daumé III. 2010. Unsupervised Part of Speech Tagging Without a Lexicon. In *NIPS Workshop on Grammar Induction, Representation of Language and Language Learning 2010*.

K. Toutanova, D. Klein, C. Manning, and Y. Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of NAACL*, pages 173–180.

J. Van Gael, A. Vlachos, and Z. Ghahramani. 2009. The infinite HMM for unsupervised PoS tagging. In *Proceedings of EMNLP*, pages 678–687.

L. Xu, D. Wilkinson, F. Southey, and D. Schuurmans. 2006. Discriminative unsupervised learning of structured predictors. In *Proceedings of ICML*, pages 1057–1064.

Q. Zhao and M. Marcus. 2009. A simple unsupervised learner for POS disambiguation rules given only a minimal lexicon. In *Proceedings of EMNLP*, pages 688–697.

Improving Gender Classification of Blog Authors

Arjun Mukherjee
Department of Computer Science
University of Illinois at Chicago
851 South Morgan Street
Chicago, IL 60607, USA
amukherj@cs.uic.edu

Bing Liu
Department of Computer Science
University of Illinois at Chicago
851 South Morgan Street
Chicago, IL 60607, USA
liub@cs.uic.edu

Abstract

The problem of automatically classifying the gender of a blog author has important applications in many commercial domains. Existing systems mainly use features such as words, word classes, and POS (part-of-speech) n-grams, for classification learning. In this paper, we propose two new techniques to improve the current result. The first technique introduces a new class of features which are variable length POS sequence patterns mined from the training data using a sequence pattern mining algorithm. The second technique is a new feature selection method which is based on an ensemble of several feature selection criteria and approaches. Empirical evaluation using a real-life blog data set shows that these two techniques improve the classification accuracy of the current state-of-the-art methods significantly.

1 Introduction

Weblogs, commonly known as blogs, refer to online personal diaries which generally contain informal writings. With the rapid growth of blogs, their value as an important source of information is increasing. A large amount of research work has been devoted to blogs in the natural language processing (NLP) and other communities. There are also many commercial companies that exploit information in blogs to provide value-added services, e.g., blog search, blog topic tracking, and sentiment analysis of people's opinions on products and services. Gender classification of blog authors is one such study, which also has many commercial applications. For example, it can help the user find what topics or products are most talked about by males and females, and what products and services are liked or disliked by men and women. Knowing this information is crucial for market intelligence because the information can be exploited in targeted advertising and also product development.

In the past few years, several authors have studied the problem of gender classification in the natural language processing and linguistic communities. However, most existing works deal with formal writings, e.g., essays of people, the Reuters news corpus and the British National Corpus (BNC). Blog posts differ from such text in many ways. For instance, blog posts are typically short and unstructured, and consist of mostly informal sentences, which can contain spurious information and are full of grammar errors, abbreviations, slang words and phrases, and wrong spellings. Due to these reasons, gender classification of blog posts is a harder problem than gender classification of traditional formal text.

Recent work has also attempted gender classification of blog authors using features such as content words, dictionary based content analysis results, POS (part-of-speech) tags and feature selection along with a supervised learning algorithm (Schler et al., 2006; Argamon et al., 2007; Yan and Yan, 2006). This paper improves these existing methods by proposing two novel techniques. The first technique adds a new class of pattern based features to learning, which are not used in any existing work. The patterns are frequent sequences of POS tags which can capture complex stylistic characteristics of male and female authors. We note that these patterns are very different from the traditional n-grams because the

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 207–217,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

patterns are of variable lengths and need to satisfy some criteria in order for them to represent significant regularities. We will discuss them in detail in Section 3.5.

The second technique is a new feature selection algorithm which uses an ensemble of feature selection criteria and methods. It is well known that each individual feature selection criterion and method can be biased and tends to favor certain types of features. A combination of them should be able to capture the most useful or discriminative features.

Our experimental results based on a real life blog data set collected from a large number of blog hosting sites show that the two new techniques enable classification algorithms to significantly improve the accuracy of the current state-of-the-art techniques (Argamon et al., 2007; Schler et al., 2006; Yan and Yan, 2006). We also compare with two publicly available systems, *Gender Genie* (BookBlog, 2007) and *Gender Guesser* (Krawetz, 2006). Both systems implemented variations of the method given in (Argamon et al., 2003). Here, the improvement of our techniques is even greater.

2 Related Work

There have been several recent papers on gender classification of blogs (e.g., Schler et al., 2006, Argamon et al., 2007; Yan and Yan, 2006; Nowson et al., 2005). These systems use function/content words, POS tag features, word classes (Schler et al., 2006), content word classes (Argamon et al., 2007), results of dictionary based content analysis, POS unigram (Yan and Yan, 2006), and personality types (Nowson et al., 2005) to capture stylistic behavior of authors' writings for classifying gender. (Koppel et al. 2002) also used POS n-grams together with content words on the British National Corpus (BNC). (Houvardas and Stamatatos, 2006) even applied character (rather than word or tag) n-grams to capture stylistic features for authorship classification of news articles in Reuters.

However, these works use only one or a subset of the classes of features. None of them uses all features for classification learning. Given the complexity of blog posts, it makes sense to apply all classes of features jointly in order to classify genders. Moreover, having many feature classes is very useful as they provide features with varied granularities and diversities. However, this also results in a huge number of features and many of them are redundant and may obscure classification. Feature selection is thus needed. Following the idea, this paper proposes a new ensemble feature selection method which is capable of extracting good features from different feature classes using multiple criteria.

We also note some less relevant literature. For example, (Tannen, 1990) deals with gender differences in "conversational style" and in "formal written essays", and (Gefen and Straub, 1997) reports differences in perception of males and females in the use of emails.

Our new POS pattern features are related to POS n-grams used in (Koppel et al., 2002; Argamon et al., 2007), which considered POS 3-grams, 2-grams and unigrams as features. As shown in (Baayen et. al. 1996), POS n-grams are very effective in capturing the fine-grained stylistic and heavier syntactic information. In this work, we go further by finding POS sequence patterns. As discussed in the introduction, our patterns are entirely different from POS n-grams. First of all, they are of variable lengths depending on whatever lengths can catch the regularities. They also need to satisfy some constraints to ensure that they truly represent some significant regularity of male or female writings. Furthermore, our POS sequence patterns can take care of n-grams and capture additional sequence regularities. These automatically mined pattern features are thus more discriminating for classification.

3 Feature Engineering and Mining

There are different *classes* of features that have been experimented for gender classification, e.g., F-measure, stylistic features, gender preferential features, factor analysis and word classes (Nowson et al., 2005; Schler et al., 2006; Corney et al., 2002; Argamon et al., 2007). We use all these existing features and also propose a new class of features that are POS sequence patterns, which replace existing POS n-grams. Also, as mentioned before, using all feature classes gives us features with varied granularities. Upon extracting all these classes of features, a new *ensemble feature selection* (EFS) algorithm is proposed to select a subset of good or discriminative features.

Below, we first introduce the existing features, and then present the proposed class of new pattern based features and how to discover them.

3.1 F-measure

The F-measure feature was originally proposed in (Heylighen and Dewaele, 2002) and has been used in (Nowson et al., 2005) with good results. Note that F-measure here is not the F-score or F-measure used in text classification or information retrieval for measuring the classification or retrieval effectiveness (or accuracy).

F-measure explores the notion of implicitness of text and is a unitary measure of text's relative contextuality (implicitness), as opposed to its formality (explicitness). Contextuality and formality can be captured by certain parts of speech. A lower score of F-measure indicates contextuality, marked by greater relative use of pronouns, verbs, adverbs, and interjections; a higher score of F-measure indicates formality, represented by greater use of nouns, adjectives, prepositions, and articles. F-measure is defined based on the frequency of the POS usage in a text (*freq.x* below means the frequency of the part-of-speech *x*):

$$F = 0.5 * [(freq.noun + freq.adj + freq.prep + freq.art) - (freq.pron + freq.verb + freq.adv + freq.int) + 100]$$

(Heylighen and Dewaele, 2002) applied the F-measure to a corpus with known author genders and found a distinct difference between the sexes. Females scored lower preferring a more contextual style while males scored higher preferring a more formal style. F-measure values for male and female writings reported in (Nowson et al., 2005) also demonstrated a similar trend. In our work, we also use F-measure as one of the features.

3.2 Stylistic Features

These are features which capture people's writing styles. The style of writing is typically captured by three types of features: part of speech, words, and in the blog context, words such as *lol*, *hmm*, and *smiley* that appear with high frequency. In this work, we use words and blog words as stylistic features. Part of speech features are mined using our POS sequence pattern mining algorithm. POS n-grams can also be used as features. However, since we mine all POS sequence patterns and use them as features, most discriminative POS n-grams are already covered. In Section 5, we will also show that POS n-grams do not perform as well as our POS sequence patterns.

3.3 Gender Preferential Features

Gender preferential features consist of a set of signals that has been used in an email gender classification task (Corney et al., 2002). These features come from various studies that have been undertaken on the issue of gender and language use (Schiffman, 2002). It was suggested by these studies and also various other works that women's language makes more frequent use of emotionally intensive adverbs and adjectives like "so", "terribly", "awfully", "dreadfully" and women's language is more punctuated. On the other hand, men's conversational patterns express "independence" (Corney et al., 2002). In brief, the language expressed by males is more proactive at solving problems while the language used by females is more reactive to the contribution of others - agreeing, understanding and supporting. We used the gender preferential features listed in Table 1, which indicate adjectives and adverbs based on the presence of suffixes and apologies as used in (Corney et al., 2002). The feature value assignment will be discussed in Section 5.

f1	words ending with *able*
f2	words ending with *al*
f3	words ending with *ful*
f4	words ending with *ible*
f5	words ending with *ic*
f6	words ending with *ive*
f7	words ending with *less*
f8	words ending with *ly*
f9	words ending with *ous*
f10	*sorry* words

Table 1: Gender preferential features

3.4 Factor Analysis and Word Classes

Factor or word factor analysis refers to the process of finding groups of similar words that tend to occur in similar documents. This process is referred to as meaning extraction in (Chung and Pennebaker, 2007). Word lists for twenty factors, along with suggested labels/headings (for reference) were used as features in (Argamon et al., 2007). Here we list some of those features (word classes) in Table 2. For the detailed list of such word classes, the reader is referred to (Argamon et al., 2007). We also used these word classes as features in our work. In addition, we added three more new word classes implying positive, negative and emotional connotations and used them as features in our experiments. These are listed in Table 3.

Factor	*Words*
Conversation	*know, people, think, person, tell, feel, friends, talk, new, talking, mean, ask, understand, feelings, care, thinking, friend, relationship, realize, question, answer, saying*
Home	*woke, home, sleep, today, eat, tired, wake, watch, watched, dinner, ate, bed, day, house, tv, early, boring, yesterday, watching, sit*
Family	*years, family, mother, children, father, kids, parents, old, year, child, son, married, sister, dad, brother, moved, age, young, months, three, wife, living, college, four, high, five, died, six, baby, boy, spend, Christmas*
Food / Clothes	*food, eating, weight, lunch, water, hair, life, white, wearing, color, ice, red, fat, body, black, clothes, hot, drink, wear, blue, minutes, shirt, green, coffee, total, store, shopping*
Romance	*forget, forever, remember, gone, true, face, spent, times, love, cry, hurt, wish, loved*

Table 2: Words in factors

Positive	*absolutely, abundance, ace, active, admirable, adore, agree, amazing, appealing, attraction, bargain, beaming, beautiful, best, better, boost, breakthrough, breeze, brilliant, brimming, charming, clean, clear, colorful, compliment, confidence, cool, courteous, cuddly, dazzling, delicious, delightful, dynamic, easy, ecstatic, efficient, enhance, enjoy, enormous, excellent, exotic, expert, exquisite, flair, free, generous, genius, great, graceful, heavenly, ideal, immaculate, impressive, incredible, inspire, luxurious, outstanding, royal, speed, splendid, spectacular, superb, sweet, sure, supreme, terrific, treat, treasure, ultra, unbeatable, ultimate, unique, wow, zest*
Negative	*wrong, stupid, bad, evil, dumb, foolish, grotesque, harm, fear, horrible, idiot, lame, mean, poor, heinous, hideous, deficient, petty, awful, hopeless, fool, risk, immoral, risky, spoil, spoiled, malign, vicious, wicked, fright, ugly, atrocious, moron, hate, spiteful, meager, malicious, lacking*
Emotion	*aggressive, alienated, angry, annoyed, anxious, careful, cautious, confused, curious, depressed, determined, disappointed, discouraged, disgusted, ecstatic, embarrassed, enthusiastic, envious, excited, exhausted, frightened, frustrated, guilty, happy, helpless, hopeful, hostile, humiliated, hurt, hysterical, innocent, interested, jealous, lonely, mischievous, miserable, optimistic, paranoid, peaceful, proud, puzzled, regretful, relieved, sad, satisfied, shocked, shy, sorry, surprised, suspicious, thoughtful, undecided, withdrawn*

Table 3: Words implying positive, negative and emotional connotations

3.5 Proposed POS Sequence Pattern Features

We now present the proposed POS sequence pattern features and the mining algorithm. This results in a new feature class. A *POS sequence pattern* is a sequence of consecutive POS tags that satisfy some constraints (discussed below). We used (Tsuruoka and Tsujii, 2005) as our POS tagger.

As shown in (Baayen et. al., 1996), POS n-grams are good at capturing the heavy stylistic and syntactic information. Instead of using all such n-grams, we want to discover all those patterns that represent true regularities, and we also want to have flexible lengths (not fixed lengths as in n-grams). POS sequence patterns serve these purposes. Its mining algorithm mines all such patterns that satisfy the user-specified minimum support (*minsup*) and minimum adherence (*minadherence*) thresholds or constraints. These thresholds ensure that the mined patterns represent significant regularities.

The main idea of the algorithm is to perform a level-wise search for such patterns, which are POS sequences with minsup and minadherence. The *support* of a pattern is simply the proportion of documents that contain the pattern. If a pattern appears too few times, it is probably spurious. A sequence is called a *frequent sequence* if it satisfies minsup. The *adherence* of a pattern is measured using the *symmetrical conditional probability* (SCP) given in (Silva et al., 1999). The SCP of a sequence with two elements $|xy|$ is the product of the conditional probability of each given the other,

$$SCP(x,y) = P(x \mid y)P(y \mid x) = \frac{P(x,y)^2}{P(x)P(y)}$$

Given a consecutive sequence of POS tags $|x_1 \ldots x_n|$, called a *POS sequence* of length n, a dispersion point defines two subparts of the sequence. A sequence of length n contains n-1 possible dispersion points. The SCP of the sequence $|x_1 \ldots x_n|$ given the dispersion point (denoted by *) $|x_1 \ldots x_{n-1} * x_n|$ is:

$$SCP((x_1 \ldots x_{n-1}), x_n) = \frac{P(x_1 \ldots x_n)^2}{P(x_1 \ldots x_{n-1})P(x_n)}$$

The SCP measure can be extended so that all possible dispersion points are accounted for.

Hence the *fair*SCP of the sequence $|x_1...x_n|$ is given by:

$$fairSCP(x_1...x_n) = \frac{P(x_1...x_n)^2}{\frac{1}{n-1}\sum_{i=1}^{n-1} P(x_1...x_i)P(x_{i+1}...x_n)}$$

*fair*SCP measures the adherence strength of POS tags in a sequence. The higher the *fair*SCP value, the more dominant is the sequence. Our POS sequence pattern mining algorithm is given below.

Input: Corpus $D = \{d \mid d$ is a document containing a sequence of POS tags$\}$, Tagset $T = \{t \mid t$ is a POS tag$\}$, and the user specified minimum support (*minsup*) and minimum adherence (*minadherence*).

Output: All POS sequence patterns (stored in *SP*) mined from *D* that satisfy *minsup* and *minadherence*.

```
Algorithm mine-POS-pats(D, T, minsup, minadherence)
1.  $C_1 \leftarrow$ count each $t$ ($\in T$) in $D$;
2.  $F_1 \leftarrow \{f \mid f \in C_1, f.\text{count} / n \geq minsup\}$;   // $n = |D|$
3.  $SP_1 \leftarrow F_1$;
4.  for ($k = 2$; $k \leq$ MAX-length; $k$++)
5.      $C_k$ = candidate-gen($F_{k-1}$);
6.      for each document $d \in D$
7.          for each candidate POS sequence $c \in C_k$
8.              if ($c$ is contained in $d$)
9.                  $c$.count++;
10.         endfor
11.     endfor
12.     $F_k \leftarrow \{c \in C_k \mid c.\text{count} / n \geq minsup\}$;
13      $SP_k \leftarrow \{f \in F_k \mid fair\text{SCP}(f) \geq minadherence\}$
14. endfor
15. return $SP \leftarrow \bigcup_k SP_k$;
```

```
Function candidate-gen($F_{k-1}$)
1. $C_k \leftarrow \varnothing$;
2. for each POS n-gram $c \in F_{k-1}$
3.     for each $t \in T$
4.         $c' \leftarrow$ addsuffix($c$, $t$);   // adds tag $t$ to $c$ as suffix
5.         add $c'$ to $C_k$;
6.     endfor
7. endfor
```

We now briefly explain the mine-POS-pats algorithm. The algorithm is based on level-wise search. It generates all POS patterns by making multiple passes over data. In the first pass, it counts the support of individual POS tags and determines which of them have minsup (line 2). Multiple occurrences of a tag in a document are counted only once. Those in F_1 are called *length 1 frequent sequences*. All length 1 sequence patterns are stored in SP_1. Since adherence is not defined for a single element, we have $SP_1 = F_1$ (line 3). In each subsequent pass *k* until MAX-length (which is the maximum length limit of the mined patterns), there are three steps:

1. Using F_{k-1} (frequent sequences found in the (*k*-1) pass) as a set of seeds, the algorithm applies candidate-gen() to generate all possibly frequent POS *k*-sequences (sequences of length *k*) (line 5). Those infrequent sequences (which are not in F_{k-1}) are discarded as adding more POS tags will not make them frequent based on the downward closure property in (Agrawal and Srikant, 1994).
2. *D* is then scanned to compute the actual support count of each candidate in C_k (lines 6-11).
3. At the end of each scan, it determines which candidate sequences have minsup and minadherence (lines 12 - 13). We compute F_k and SP_k separately because adherence does not have the downward closure property as the support.

Finally, the algorithm returns the set of all sequence patterns (line 15) that meet the minsup and minadherence thresholds.

The candidate-gen() function generates all possibly frequent *k*-sequences by adding each POS tag *t* to *c* as suffix. *c* is a *k*-1-sequence in F_{k-1}.

In our experiments, we used MAX-length = 7, *minsup* = 30%, and *minadherence* = 20% to mine all POS sequence patterns. All the mined patterns are used as features.

Finally, it is worthwhile to note that mine-POS-pat is very similar to the well-known GSP algorithm (Srikant and Agrawal, 1996). Likewise, it has linear scale up with data size. If needed, one can use MapReduce (Dean and Ghemawat, 2004) with suitable modifications in mine-POS-pats to speed things up by distributing to multiple machines for large corpora. Moreover, mining is a part of preprocessing of the algorithm and its complexity does not affect the final prediction, as it will be later shown that for model building and prediction, standard machine learning methods are used.

4 Ensemble Feature Selection

Since all classes of features discussed in Section 3 are useful, we want to employ all of them. This results in a huge number of features. Many of them are redundant and even harmful. Feature selection thus becomes important. There are two common approaches to feature selection: the *filter* and the *wrapper* approaches (Blum and Langley, 1997; Kohavi and John, 1997). In the filter approach, features are first ranked based on a feature selection criterion such as information gain, chi-square (χ^2) test, and mutual information. A set of top ranked features are selected. On the contrary, the wrapper model chooses features and adds to the current feature pool based on whether the new features improve the classification accuracy.

Both these approaches have drawbacks. While the wrapper approach becomes very time consuming and impractical when the number of features is large as each feature is tested by building a new classifier. The filter approach often uses only one feature selection criterion (e.g., information gain, chi-square, or mutual information). Due to the bias of each criterion, using only a single one may result in missing out some good features which can rank high based on another criterion. In this work, we developed a novel feature selection method that uses multiple criteria, and combines both the wrapper and the filter approaches. Our method is called *ensemble feature selection* (EFS).

4.1 EFS Algorithm

EFS takes the best of both worlds. It first uses a number of feature selection criteria to rank the features following the filter model. Upon ranking, the algorithm generates some candidate feature subsets which are used to find the final feature set based on classification accuracy using the wrapper model. Since our framework generates much fewer candidate feature subsets than the total number of features, using wrapper model with candidate feature sets is scalable. Also, since the algorithm generates candidate feature sets using multiple criteria and all feature classes jointly, it is able to capture most of those features which are discriminating. We now detail our EFS algorithm.

The algorithm takes as input, a set of n features $F = \{f_1, \ldots, f_n\}$, a set of t feature selection criteria $\Theta = \{\theta_1, \ldots, \theta_t\}$, a set of t thresholds $T = \{\tau_1, \ldots, \tau_t\}$ corresponding to the criteria in Θ, and a window w. τ_i is the base number of features to be selected for criterion θ_i. w is used to vary τ_i (thus the number of features) to be used by the wrapper approach.

Algorithm: EFS (F, Θ, T, w)

```
1.  for each $\theta_i \in \Theta$
2.      Rank all features in $F$ based on criterion $\theta_i$ and
          let $\xi_i$ denotes the ranked features
3.  endfor
4.  for $i = 1$ to $t$
5.      $C_i \leftarrow \varnothing$
6.      for $\tau = \tau_i - w$ to $\tau = \tau_i + w$
7.          select first $\tau$ features $\zeta_i$ from $\xi_i$ and add $\zeta_i$ to $C_i$
              in order
8.      endfor
9.  endfor
10.     // $C_i = \{\zeta_1, \ldots, \zeta_{2w+1}\}$, where $\zeta_i$ is a set of fea-
        tures
11. OptCandFeatures $\leftarrow \varnothing$;
12. Repeat steps 13 – 18
13.     $\Lambda \leftarrow \varnothing$
14.     for $i = 1$ to $t$
15.         select and remove the *first* feature set $\zeta_i \in C_i$
              from $C_i$ in order
16.         $\Lambda \leftarrow \Lambda \cup \zeta_i$
17.     endfor
18.     add $\Lambda$ to OptCandFeatures
19.     // $\Lambda$ is a set of features comprising of features in
        // feature sets $\zeta_i \in C_i$ in the same position $\forall\ i$
20. until $C_i = \varnothing\ \forall\ i$
21. for each $\Lambda \in$ OptCandFeatures
22.     $\Lambda$.score $\leftarrow$ accuracy of 10-fold CV on training
          data on a chosen classifier (learning algo-
          rithm)
23. endfor
24. return $\underset{\Lambda.score}{\arg\max}\{\Lambda \mid \Lambda \in \text{OptCandFeatures}\}$
```

We now explain our EFS algorithm. Using a set of different feature selection measures, Θ, we rank all features in our feature pool, F, using the set of criteria (lines 1–3). This is similar to the filter approach. In lines 4–9, we generate feature sets C_i, $1 \leq i \leq t$ for each of the t criteria. Each set C_i contains feature subsets, and each subset ζ_i is the set of top τ features in ξ_i ranked based on criterion θ_i in lines 1–2. τ varies from $\tau_i - w$ to $\tau_i + w$ where τ_i is the threshold for criterion θ_i and w the window size. We vary τ and generate $2w + 1$ feature sets and add all such feature sets ζ_i to C_i (in lines 6–8) in order. We do so because it is difficult to know the optimal threshold τ_i for each criterion θ_i. It should be noted that "adding in order" ensures the ordering of feature sets ζ_i as shown in line 10, which will be later used to "select and remove in order" in line 15. In lines 11–20 we generate candidate feature sets using C_i and add each such candidate feature set Λ to OptCandFeatures. Each candidate feature set Λ is a collection of top ranked features based on multiple criteria. It is generated by unioning the features in the *first* feature subset ζ_i, which is then removed from C_i for each criterion θ_i (lines 14-17). Each candidate feature set is added to OptCandFeatures in line 18. Since each C_i has $2w$+1 feature subsets ζ_i, there are a total of $2w$+1 candidate feature sets Λ in OptCandFeatures. Lines 21–23 assign an accuracy to each candidate feature set Λ ϵ OptCandFeatures by running 10-fold cross validation on the training data using a chosen classifier with the features in Λ. Finally, the optimal feature set Λ ϵ OptCandFeatures is returned in line 24.

An interesting question arising in the EFS algorithm is: How does one select the threshold τ_i for each criterion θ_i and the window size w? Intuitively, suppose that for criterion θ_i, the optimal subset of features is S_{opt_i} based on some optimal threshold τ_i. Then the final feature set is a collection of all features $f \in S_{opt\_i}$ $\forall$ i. However, finding such optimal feature set S_{opt_i} or optimal threshold τ_i is a difficult problem. To counter this, we use the window w to select various feature subsets close to the top τ_i features in ξ_i. Thus, the threshold values τ_i and window size w should be approximated by experiments. In our experiments, we used τ_i = top 1/20th of the features ranked in ξ_i for $\forall$ i and window size $w = |F|/100$, and got good results. Fortunately, as we will see in Section 6.2, these parameters are not sensitive at all, and any reasonably large size feature set seems to work equally well.

Finally, we are aware that there are some existing ensemble feature selection methods in the machine learning literature (Garganté et al., 2007; Tuv et al., 2009). However, they are very different from our approach. They mainly use ensemble classification methods to help choose good features rather than combining different feature selection criteria and integrating different feature selection approaches as in our method.

4.2 Feature Selection Criteria

The set of feature selection criteria $\Theta = \{\theta_1 \ldots \theta_t\}$ used in our work are those commonly used individual selection criteria in the filter approach.

Let $C = \{c_1, c_2, \ldots, c_m\}$ denotes the set of classes, and $F = \{f_1, f_2, \ldots, f_n\}$ the set of features. We list the criteria in Θ used in our work below.

Information Gain (IG): This is perhaps the most commonly used criterion, which is based on entropy. The scoring function for information gain of a feature f is given by:

$$IG(f) = -\sum_{i=1}^{m} P(c_i)\log P(c_i) + \sum_{f,\bar{f}} P(f) \sum_{i=1}^{m} P(c_i \mid f)\log P(c_i \mid f)$$

Mutual Information (MI): This metric is commonly used in statistical language modeling. The mutual information MI(f, c) between a class c and a feature f is defined as:

$$MI(f,c) = \sum_{f,\bar{f}} \sum_{c,\bar{c}} P(f,c) \log \frac{P(f,c)}{P(f)P(c)}$$

The scoring function generally used as the criterion is the max among all classes. MI(f) = max$_i$ {MI (f, c_i)} (which we use). The weighted average over all classes can also be applied as the scoring function.

χ^2 Statistic: The χ^2 statistic measures the lack of independence between a feature f and class c, and can be compared to the χ^2 distribution with one degree of freedom. We use a 2x2 contingency table of a feature f and a class c to introduce χ^2 test.

	c	$\bar{c}$
f	W	X
$\bar{f}$	Y	Z

Table 4: Two-way contingency table of f and c

In the table, W denotes the number of documents in the corpus in which feature f and class c co-occur, X the number of documents in which f occurs without c, Y the number of documents in which c occurs without f, and Z the number of documents in which neither c nor f occurs. Thus, $N = W + X + Y + Z$ is the total number of documents in the corpus.

χ^2 test is defined as:

$$\chi^2(f,c) = \frac{N(WZ - YX)^2}{(W+Y)(X+Z)(W+X)(Y+Z)}$$

The scoring function using the χ^2 statistic is either the weighted average or max over all classes. In our experiments, we use the weighted average: $\chi^2(f) = \sum_{i=1}^{m} P(c_i)\chi^2(f,c_i)$

Cross Entropy (CE): This metric is similar to mutual information (Mladenic and Grobelnik, 1998):

$$CE(f) = P(f)\sum_{i=1}^{m} P(c_i \mid f)\log\frac{P(c_i \mid f)}{P(f)}$$

Weight of Evidence for Text (WET): This criterion is based on the average absolute weight of evidence (Mladenic and Grobelnik, 1998):

$$WET(f) = \sum_{i=1}^{m} P(c_i)P(f) \mid \log\frac{P(c_i \mid f)(1-P(c_i))}{P(c_i)(1-P(c_i \mid f))} \mid$$

5 Feature Value Assignments

After selecting features belonging to different classes, values are assigned differently to different classes of features. There are three common ways of feature value assignments: Boolean, TF (Term Frequency) and TF-IDF (product of term and inverted document frequency). For details of feature value assignments, interested readers are referred to (Joachims, 1997). While the Boolean scheme assigns a 1 to the feature value if the feature is present in the document and a 0 otherwise, the TF scheme assigns the relative frequency of the number of times that the feature occurs in the document. We did not use TF-IDF as it did not yield good results in our preliminary experiments.

The feature value assignment to different classes of features is done as follows: The value of F-measure was assigned based on its actual value. Stylistic features such words, and blog words were assigned values 1 or 0 in the Boolean scheme and the relative frequency in the TF scheme (we experimented with both schemes). Feature values for gender preferential features were also assigned in a similar way. Factor and word class features were assigned values according to the Boolean or TF scheme if any of the words belonging to the feature class exists (factor or word class appeared in that document). Each POS sequence pattern feature was assigned a value according to the Boolean (or TF) scheme based on the appearances of the pattern in the POS tagged document.

6 Experimental Results

This section evaluates the proposed techniques and sees how they affect the classification accuracy. We also compare with the existing state-of-the-art algorithms and systems. For algorithms, we compared with three representatives in (Argamon et al., 2007), (Schler et al., 2006) and (Yan and Yan, 2006). Since they do not have publicly available systems, we implemented them. Each of them just uses a subset of the features used in our system. Recall our system includes all their features and our own POS pattern based features. For systems, we compared with two public domain systems, *Gender Genie* (BookBlog, 2007) and *Gender Guesser* (Krawetz, 2006), which implemented variations of the algorithm in (Argamon et. al, 2003).

We used SVM classification, SVM regression, and Naïve Bayes (NB) as learning algorithms. Although SVM regression is not designed for classification, it can be applied based on the output of positive or negative values. It actually worked better than SVM classification for our data. For SVM classification and regression, we used SVMLight (Joachims, 1999), and for NB we used (Borgelt, 2003). In all our experiments, we used accuracy as the evaluation measure as the two classes (male and female) are roughly balanced (see the data description below), and both classes are equally important.

6.1 Blog Data Set

To keep the problem of gender classification of informal text as general as possible, we collected blog posts from many blog hosting sites and blog search engines, e.g., blogger.com, technorati.com, etc. The data set consists of 3100 blogs. Each blog is labeled with the gender of its author. The gender of the author was determined by visiting the profile of the author. Profile pictures or avatars associated with the profile were also helpful in confirming the gender especially when the gender information was not available explicitly. To ensure quality of the labels, one group of students collected the blogs and did the initial labeling, and the other group double-checked the labels by visiting the actual blog pages. Out of 3100 posts, 1588 (51.2%) were written by men and 1512 (48.8%) were written by women. The average post length is 250 words for men and 330 words for women.

6.2 Results

We used all features from different feature classes (Section 3) along with our POS patterns as our pool of features. We used τ and w values stated in Section 4.1 and criteria mentioned in Section 4.2 for our EFS algorithm. EFS was compared with three commonly used feature selection methods on SVM classification (denoted by SVM), SVM regression (denoted by SVM_R) and the NB classifier. The results are shown in Table 5. All results were obtained through 10-fold cross validation.

Also, the total number of features selected by IG, MI, χ^2, and EFS were roughly the same. Thus, the improvement in accuracy brought forth by EFS was chiefly due to the combination of features selected (based on multi-criteria).

To measure the accuracy improvement of using our POS patterns over common POS n-grams, we also compared our results with those from POS n-grams (Koppel et al., 2002). The comparison results are given in Table 6. Table 6 also includes results to show the overall improvement in accuracy with our two new techniques. We tested our system without any feature selection and without using the POS sequence patterns as features.

The comparison results with existing algorithms and public domain systems using our real-life blog data set are tabulated in Table 7.

Also, to see whether feature selection helps and how many features are optimal, we varied τ and w of the EFS algorithm and plotted the accuracy vs. no. of features. These results are shown in Figure 1.

Feature Selection	Value Assignment	NB	SVM	SVM_R
IG	Boolean	71.32	76.61	78.32
IG	TF	66.01	72.84	74.13
MI	Boolean	72.01	78.62	79.48
MI	TF	70.86	73.14	74.58
χ^2	Boolean	72.90	80.71	81.52
χ^2	TF	71.84	73.57	75.24
EFS	Boolean	73.57	86.24	88.56
EFS	TF	72.82	82.05	83.53

Table 5: Accuracies of SVM, SVM_R and NB with different feature selection methods

Settings	NB	SVM	SVM_R
All features	63.01	68.84	70.03
All features, no POS patterns	60.73	65.17	66.17
POS 1,2,3-grams + EFS	71.24	82.71	83.86
POS Patterns + EFS	73.57	86.24	88.56

Table 6: Accuracies of POS n-grams and POS patterns with or without EFS (Boolean value assignment)

System	Accuracy (%)
Gender Genie	61.69
Gender Guesser	63.78
(Argamon et al., 2007)	77.86
(Schler et al., 2006)	79.63
(Yan and Yan, 2006)	68.75
Our method	88.56

Table 7: Accuracy comparison with other systems

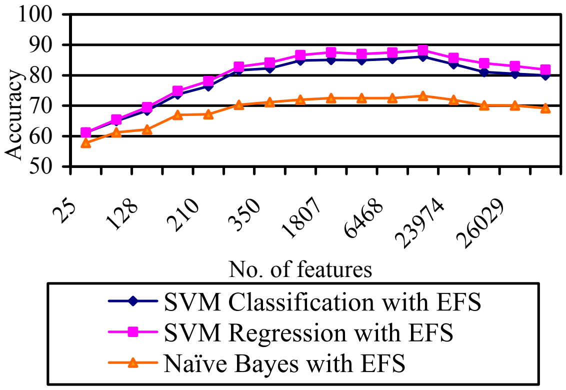


Figure 1: Accuracy vs. no. of features using EFS

6.3 Observations and Discussions

Based on the results given in the previous section, we make the following observations:

- SVM regression (SVM_R) performs the best (Table 5). SVM classification (SVM) also gives good accuracies. NB did not do so well.
- Table 5 also shows that our EFS feature selection method brings about 6-10% improvement in accuracy over the other feature selection methods based on SVM classification and SVM regression. The reason has been explained in the introduction section. Paired *t*-tests showed that all the improvements are statistically significant at the confidence level of 95%. For NB, the benefit is less (3%).
- Keeping all other parameters constant, Table 5 also shows that Boolean feature values yielded better results than the TF scheme across all classifiers and feature selection methods.
- Row 1 of Table 6 tells us that feature selection is very useful. Without feature selection (All features), SVM regression only achieves 70% accuracy, which is way inferior to the 88.56% accuracy obtained using EFS feature selection. Row 2 shows that without EFS and without POS sequence patterns, the results are even worse.

- Keeping all other parameters intact, Table 6 also demonstrated the effectiveness of our POS pattern features over POS n-grams. We have discussed the reason in Section 3.2 and 3.5.
- From Tables 5 and 6, we can infer that the overall accuracy improvement using EFS and all feature classes described in Section 3 is about 15% for SVM classification and regression and 10% for NB. Also, using POS sequence patterns with EFS brings about a 5% improvement over POS n-grams (Table 6). The improvement is more pronounced for SVM based methods than NB.
- Table 7 summarizes the accuracy improvement brought by our proposed techniques over the existing state-of-art systems. Our techniques have resulted in substantial (around 9%) accuracy improvement over the best of the existing systems. Note that (Argamon et al., 2007) used Logistic Regression with word classes and POS unigrams as features. (Schler et al., 2006) used Winnow classifier with function words, content word classes, and POS features. (Yan and Yan, 2006) used Naive Bayes with content words and blog-words as features. For all these systems, we used their features and ran their original classifiers and also the three classifiers in this paper and report the best results. For example, for (Argamon et al., 2007), we ran Logistic Regression and our three methods. SVM based methods always gave slightly better results. We could not run Winnow due to some technical issues. SVM and SVM_R gave comparable results to those given in their original papers. These results again show that our techniques are useful. All the gains are statistically significant at the confidence level of 95%.
- From Figure 1, we see that when the number of features selected is small (<100) the classification accuracy is lower than that obtained by using all features (no feature selection). However, the accuracy increases rapidly as the number of selected features increases. After obtaining the best case accuracy, it roughly maintains the accuracy over a long range. The accuracies then gradually decrease with the increase in the number of features. This trend is consistent with the prior findings in (Mladenic, 1998; Rogati and Yang, 2002; Forman 2003; Riloff et al., 2006; Houvardas and Stamatatos, 2006).

  It is important to note here that over a long range of 2000 to 20000 features, the accuracy is high and stable. This means that the thresholds of EFS are easy to set. As long as they are in the range, the accuracy will be good.

Finally, we would like to mention that (Herring and Paolillo, 06) has used genre relationships with gender classification. Their finding that subgenre “diary” contains more “female” and subgenre “filter” having more “male” stylistic features independent of the author gender, may obscure gender classification as there are many factors to be considered. Herring and Paolillo referred only words as features which are not as fine grained as our POS sequence patterns. We are also aware of other factors influencing gender classification like genre, age and ethnicity. However, much of such information is hard to obtain reliably in blogs. They definitely warren some future studies. Also, EFS being a useful method for feature selection in machine learning, it would be useful to perform further experiments to investigate how well it performs on a variety of classification datasets. This again will be an interesting future work.

7 Conclusions

This paper studied the problem of gender classification. Although there have been several existing papers studying the problem, the current accuracy is still far from ideal. In this work, we followed the supervised approach and proposed two novel techniques to improve the current state-of-the-art. In particular, we proposed a new class of features which are POS sequence patterns that are able to capture complex stylistic regularities of male and female authors. Since there are a large number features that have been considered, it is important to find a subset of features that have positive effects on the classification task. Here, we proposed an ensemble feature selection method which takes advantage of many different types of feature selection criteria in feature selection. Experimental results based on a real-life blog data set demonstrated the effectiveness of the proposed techniques. They help achieve significantly higher accuracy than the current state-of-the-art techniques and systems.

References

Agrawal, R. and Srikant, R. 1994. *Fast Algorithms for Mining Association Rules*. VLDB. pp. 487-499.

Argamon, S., Koppel, M., J Fine, AR Shimoni. 2003. *Gender, genre, and writing style in formal written texts*. Text-Interdisciplinary Journal, 2003.

Argamon, S., Koppel, M., Pennebaker, J. W., Schler, J. 2007. *Mining the Blogosphere: Age, Gender and the varieties of self-expression*, First Monday, 2007 - firstmonday.org

Baayen, H., H van Halteren, F Tweedie. 1996. *Outside the cave of shadows: Using syntactic annotation to enhance authorship attribution*, Literary and Linguistic Computing, 11, 1996.

Blum, A. and Langley, P. 1997. *Selection of relevant features and examples in machine learning*. Artificial Intelligence, 97(1-2):245-271.

BookBlog, *Gender Genie*, Copyright 2003-2007, http://www.bookblog.net/gender/genie.html

Borgelt, C. 2003. *Bayes Classifier Induction*. http://www.borgelt.net/doc/bayes/bayes.html

Chung, C. K. and Pennebaker, J. W. 2007. *Revealing people's thinking in natural language: Using an automated meaning extraction method in open–ended self–descriptions*, J. of Research in Personality.

Corney, M., Vel, O., Anderson, A., Mohay, G. 2002. *Gender Preferential Text Mining of E-mail Discourse*. 18th annual Computer Security Applications Conference (ACSAC), 2002.

J. Dean and S. Ghemawat. 2004. *Mapreduce: Simplified data processing on large clusters*, Operating Systems Design and Implementation, 2004.

Forman, G., 2003. *An extensive empirical study of feature selection metrics for text classification.* JMLR, 3:1289 - 1306 , 2003.

Garganté, R. A., Marchiori, T. E., and Kowalczyk, S. R. W., 2007. *A Genetic Algorithm to Ensemble Feature Selection*. Masters Thesis. Vrije Universiteit, Amsterdam.

Gefen, D., D. W. Straub. 1997. *Gender differences in the perception and use of e-mail: An extension to the technology acceptance model*. MIS Quart. 21(4) 389–400.

Herring, S. C., & Paolillo, J. C. 2006. *Gender and genre variation in weblogs*, Journal of Sociolinguistics, 10 (4), 439-459.

Heylighen, F., and Dewaele, J. 2002. *Variation in the contextuality of language*: an empirical measure. Foundations of Science, 7, 293–340.

Houvardas, J. and Stamatatos, E. 2006. *N-gram Feature Selection for Authorship Identification*, Proc. of the 12th Int. Conf. on Artificial Intelligence: Methodology, Systems, Applications, pp. 77-86.

Joachims, T. 1999. *Making large-Scale SVM Learning Practical*. Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.

Joachims, T. 1997. *Text categorization with support vector machines*, Technical report, LS VIII Number 23, University of Dortmund, 1997

Kohavi, R. and John, G. 1997. *Wrappers for feature subset selection*. Artificial Intelligence, 97(1-2):273-324.

Koppel, M., Argamon, S., Shimoni, A. R.. 2002. *Automatically Categorizing Written Text by Author Gender*. Literary and Linguistic Computing.

Krawetz, N. 2006. *Gender Guesser*. Hacker Factor Solutions. http://www.hackerfactor.com/ GenderGuesser.html

Mladenic, D. 1998. *Feature subset selection in text learning*. In Proc. of ECML-98, pp. 95–100.

Mladenic, D. and Grobelnik, D.1998. *Feature selection for classification based on text hierarchy*. Proceedings of the Workshop on Learning from Text and the Web, 1998

Nowson, S., Oberlander J., Gill, A. J., 2005. *Gender, Genres, and Individual Differences*. In Proceedings of the 27th annual meeting of the Cognitive Science Society (p. 1666–1671). Stresa, Italy.

Riloff, E., Patwardhan, S., Wiebe, J.. 2006. *Feature Subsumption for opinion Analysis*. EMNLP,

Rogati, M. and Yang, Y.2002. *High performing and scalable feature selection for text classification*. In CIKM, pp. 659-661, 2002.

Schiffman, H. 2002. Bibliography of Gender and Language. http://ccat.sas.upenn.edu/~haroldfs/ popcult/bibliogs/gender/genbib.htm

Schler, J., Koppel, M., Argamon, S, and Pennebaker J. 2006. *Effects of age and gender on blogging*, In Proc. of the AAAI Spring Symposium Computational Approaches to Analyzing Weblogs.

Silva, J., Dias, F., Guillore, S., Lopes, G. 1999. *Using LocalMaxs Algortihm for the Extraction of Contiguous and Noncontiguous Multiword Lexical Units*. Springer Lecture Notes in AI 1695, 1999

Srikant, R. and Agrawal, R. 1996. *Mining sequential patterns: Generalizations and performance improvements*, In Proc. 5th Int. Conf. Extending Database Technology (EDBT'96), Avignon, France.

Tannen, D. (1990). *You just don't understand*, New York: Ballantine.

Tsuruoka, Y. and Tsujii, J. 2005. *Bidirectional Inference with the Easiest-First Strategy for Tagging Sequence Data*, HLT/EMNLP 2005, pp. 467-474.

Tuv, E., Borisov, A., Runger, G., and Torkkola, K. 2009. *Feature selection with ensembles, artificial variables, and redundancy elimination.* JMLR, 10.

Yan, X., Yan, L. 2006. *Gender Classification of Weblog Authors*. Computational Approaches to Analyzing Weblogs, AAAI.

Negative Training Data can be Harmful to Text Classification

Xiao-Li Li
Institute for Infocomm Research
1 Fusionopolis Way #21-01,
Connexis Singapore 138632
xlli@i2r.a-star.edu.sg

Bing Liu
University of Illinois at Chicago
851 South Morgan Street,
Chicago, IL 60607-7053, USA
liub@cs.uic.edu

See-Kiong Ng
Institute for Infocomm Research
1 Fusionopolis Way #21-01,
Connexis Singapore 138632
skng@i2r.a-star.edu.sg

Abstract

This paper studies the effects of training data on binary text classification and postulates that negative training data is not needed and may even be harmful for the task. Traditional binary classification involves building a classifier using labeled positive and negative training examples. The classifier is then applied to classify test instances into positive and negative classes. A fundamental assumption is that the training and test data are identically distributed. However, this assumption may not hold in practice. In this paper, we study a particular problem where the positive data is identically distributed but the negative data may or may not be so. Many practical text classification and retrieval applications fit this model. We argue that in this setting negative training data should not be used, and that PU learning can be employed to solve the problem. Empirical evaluation has been conducted to support our claim. This result is important as it may fundamentally change the current binary classification paradigm.

1 Introduction

Text classification is a well-studied problem in machine learning, natural language processing, and information retrieval. To build a text classifier, a set of training documents is first labeled with predefined classes. Then, a supervised machine learning algorithm (e.g., Support Vector Machines (SVM), naïve Bayesian classifier (NB)) is applied to the training examples to build a classifier that is subsequently employed to assign class labels to the instances in the test set. In this paper, we focus on binary text classification with two classes (i.e. *positive* and *negative* classes).

Most learning methods assume that the training and test data have identical distributions. However, this assumption may not hold in practice, i.e., the training and the test distributions can be different. The problem is called *covariate shift* or *sample selection bias* (Heckman 1979; Shimodaira 2000; Zadrozny 2004; Huang et al. 2007; Sugiyama et al. 2008; Bickel et al. 2009). In general, this problem is not solvable because the two distributions can be arbitrarily far apart from each other. Various assumptions were made to solve special cases of the problem. One main assumption was that the conditional distribution of the class given an instance is the same over the training and test sets (Shimodaira 2000; Huang et al. 2007; Bickel et al. 2009).

In this paper, we study another special case of the problem in which the positive training and test samples have identical distributions, but the negative training and test samples may have different distributions. We believe this scenario is more applicable for binary text classification. As the focus in many applications is on identifying positive instances correctly, it is important that the positive training and the positive test data have the same distribution. The distributions of the negative training and negative test data can be different. We believe that this special case of the sample selection bias problem is also more applicable for machine learning. We will show that a partially supervised learning model, called *PU learning* (learning from *P*ositive and *U*nlabeled examples) fits this special case quite well (Liu et al. 2002).

Following the notations in (Bickel et al. 2009), our special case of the sample selection bias problem can be formulated as follows: We are given a training sample matrix $\mathbf{X}_L$ with row vectors $\mathbf{x}_1$, …, $\mathbf{x}_k$. The positive and negative training instances are governed by different unknown distributions $p(\mathbf{x}|\lambda)$

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 218–228,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

and $p(\mathbf{x}|\delta)$ respectively. The element y_i of vector $\mathbf{y} = (y_1, y_2, \ldots, y_k)$ is the class label for training instance x_i ($y_i \in \{+1, -1\}$, where +1 and -1 denote positive and negative classes respectively) and is drawn based on an unknown target concept $p(y|\mathbf{x})$. In addition, we are also given an unlabeled test set in matrix $\mathbf{X}_T$ with rows $\mathbf{x}_{k+1}, \ldots, \mathbf{x}_{k+m}$. The (hidden) positive test instances in $\mathbf{X}_T$ are also governed by the unknown distribution $p(\mathbf{x}|\lambda)$, but the (hidden) negative test instances in $\mathbf{X}_T$ are governed by an unknown distribution, $p(\mathbf{x}|\theta)$, where θ may or may not be the same as δ. $p(\mathbf{x}|\theta)$ and $p(\mathbf{x}|\delta)$ can differ arbitrarily, but there is only one unknown target conditional class distribution $p(y|\mathbf{x})$.

This problem setting is common in many applications, especially in those applications where the user is interested in identifying a particular type of documents (i.e. binary text classification). For example, we want to find sentiment analysis papers in the literature. For training a text classifier, we may label the papers in some EMNLP proceedings as sentiment analysis (positive) and non-sentiment analysis (negative) papers. A classifier can then be built to find sentiment analysis papers from ACL and other EMNLP proceedings. However, this labeled training set will not be appropriate for identifying sentiment analysis papers from the WWW, KDD and SIGIR conference proceedings. This is because although the sentiment analysis papers in these proceedings are similar to those in the training data, the non-sentiment analysis papers in these conferences can be quite different. Another example is email spam detection. A spam classification system built using the training data of spam and non-spam emails from a university may not perform well in a company. The reason is that although the spam emails (e.g., unsolicited commercial ads) are similar in both environments, the non-spam emails in them can be quite different.

One can consider labeling the negative data in each environment individually so that only the negative instances relevant to the testing environment are used to train the classifier. However, it is often impractical (if not impossible) to do so. For example, given a large blog hosting site, we want to classify its blogs into those that discuss stock markets (positive), and those that do not (negative). In this case, the negative data covers an arbitrary range of topics. It is clearly impractical to label all the negative data.

Most existing methods for addressing the sample selection bias problem work as follows. First, they estimate the bias of the training data based on the given test data using statistical methods. Then, a classifier is trained on a weighted version of the original training set based on the estimated bias. In this paper, we show that our special case of the sample selection bias problem can be solved in a much simpler and somewhat radical manner—by simply discarding the negative training data altogether. We can use the positive training data and the unlabeled test data to build the classifier using the PU learning model (Liu et al. 2002).

PU learning was originally proposed to solve the learning problem where no labeled negative training data exist. Several algorithms have been developed in the past few years that can learn from a set of labeled positive examples augmented with a set of unlabeled examples. That is, given a set P of positive examples of a particular class (called the *positive* class) and a set U of unlabeled examples (which contains both hidden positive and hidden negative examples), a classifier is built using P and U to classify the data in U as well as future test data into two classes, i.e., those belonging to P (positive) and those not belonging to P (negative). In this paper, we also propose a new PU learning method which gives more consistently accurate results than the current methods.

Our experimental evaluation shows that when the distributions of the negative training and test samples are different, PU learning is much more accurate than traditional supervised learning from the positive and negative training samples. This means that the negative training data actually harms classification in this case. In addition, when the distributions of the negative training and test samples are identical, PU learning is shown to perform equally well as supervised learning, which means that the negative training data is not needed.

This paper thus makes three contributions. First, it formulates a new special case of the sample selection bias problem, and proposes to solve the problem using PU learning by discarding the negative training data. Second, it proposes a new PU learning method which is more accurate than the existing methods. Third, it experimentally demonstrates the effectiveness of the proposed method and shows that negative training data is not needed and can even be harmful. This result is important as it may fundamentally change the way that many practical classification problems should be solved.

2 Related Work

A key assumption made by most machine learning algorithms is that the training and test samples must be drawn from the same distribution. As mentioned, this assumption can be violated in practice. Some researchers have addressed this problem under *covariate shift* or *sample selection bias*. Sample selection bias was first introduced in the econometrics by Heckman (1979). It came into the field of machine learning through the work of Zadrozny (2004). The main approach in machine learning is to first estimate the distribution bias of the training data based on the test data, and then learn using weighted training examples to compensate for the bias (Bickel et al. 2009).

Shimodaira (2000) and Sugiyama and Muller (2005) proposed to estimate the training and test data distributions using kernel density estimation. The estimated density ratio could then be used to generate weighted training examples. Dudik et al. (2005) and Bickel and Scheffer (2007) used maximum entropy density estimation, while Huang et al. (2007) proposed kernel mean matching. Sugiyama et al. (2008) and Tsuboi et al. (2008) estimated the weights for the training instances by minimizing the Kullback-Leibler divergence between the test and the weighted training distributions. Bickel et al. (2009) proposed an integrated model. In this paper, we adopt an entirely different approach by dropping the negative training data altogether in learning. Without the negative training data, we use PU learning to solve the problem (Liu et al. 2002; Yu et al. 2002; Denis et al. 2002; Li et al. 2003; Lee and Liu, 2003; Liu et al. 2003; Denis et al. 2003; Li et al. 2007; Elkan and Noto, 2008; Li et al. 2009; Li et al. 2010). We will discuss this learning model further in Section 3.

Another related work to ours is transfer learning or domain adaptation. Unlike our problem setting, transfer learning addresses the scenario where one has little or no training data for the target domain, but has ample training data in a related domain where the data could be in a different feature space and follow a different distribution. A survey of transfer learning can be found in (Pan and Yang 2009). Several NLP researchers have studied transfer learning for different applications (Wu et al. 2009a; Yang et al. 2009; Agirre & Lacalle 2009; Wu et al. 2009b; Sagae & Tsujii 2008; Goldwasser & Roth 2008; Li and Zong 2008; Andrew et al. 2008; Chan and Ng 2007; Jiang and Zhai 2007; Zhou et al. 2006), but none of them addresses the problem studied here.

3 PU Learning Techniques

In traditional supervised learning, ideally, there is a large number of labeled positive and negative examples for learning. In practice, the negative examples can often be limited or unavailable. This has motivated the development of the model of *learning from positive and unlabeled examples*, or PU learning, where *P* denotes a set of positive examples, and *U* a set of unlabeled examples (which contains both hidden positive and hidden negative instances). The PU learning problem is to build a classifier using *P* and *U* in the absence of negative examples to classify the data in *U* or a future test data *T*. In our setting, the test set *T* will also act as the unlabeled set *U*.

PU learning has been investigated by several researchers in the past decade. A study of PAC learning for the setting under the statistical query model was given in (Denis, 1998). Liu et al. reported the sample complexity result and showed how the problem may be solved (Liu et al., 2002). Subsequently, a number of practical algorithms (e.g., Liu et al., 2002; Yu et al., 2002; Li and Liu, 2003) were proposed. They generally follow a two-step strategy: (i) identifying a set of *reliable negative* documents *RN* from the unlabeled set; and then (ii) building a classifier using *P* (positive set), *RN* (reliable negative set) and *U-RN* (unlabelled set) by applying an existing learning algorithm (such as naive Bayesian classifier or SVM) iteratively. There are also some other approaches based on unbalanced errors (e.g., Liu et al. 2003; Lee and Liu, 2003; Elkan and Noto, 2008).

In this section, we first introduce a representative PU learning technique S-EM, and then present a new technique called CR-SVM.

3.1 S-EM Algorithm

S-EM (Liu et al. 2002) is based on naïve Bayesian classification (NB) (Lewis, 1995; Nigam et al., 2000) and the EM algorithm (Dempster et al. 1977). It has two steps. The first step uses a *spy* technique to identify some *reliable negatives* (*RN*) from the unlabeled set *U* and the second step uses the EM algorithm to learn a Bayesian classifier from *P*, *RN* and *U–RN*.

Step 1: Extracting reliable negatives *RN* from *U* using a spy technique

The *spy* technique in S-EM works as follows (Figure 1): First, a small set of positive examples (denoted by *SP*) called "spies" is randomly sampled from *P* (line 2). The default sampling ratio in S-EM is s = 15%. Then, an NB classifier is built using *P–SP* as the positive set and $U \cup SP$ as the negative set (lines 3-5). The NB classifier is applied to classify each $u \in U \cup SP$, i.e., to assign a probabilistic class label $p(+|u)$ (+ means positive) to u. The idea of the spy technique is as follows. Since the spy examples were from *P* and were put into *U* as negatives in building the NB classifier, they should behave similarly to the hidden positive instances in *U*. We thus can use them to find the reliable negative set *RN* from *U*. Using the probabilistic labels of spies in *SP* and an input parameter l (noise level), a probability threshold t is determined. Due to space constraints, we are unable to explain l. Details can be found in (Liu et al. 2002). t is then used to find *RN* from *U* (lines 8-10).

```
1.  $RN \leftarrow \varnothing$;                       // Reliable negative set
2.  $SP \leftarrow Sample(P, s\%)$;            // spy set
3.  Assign each example in $P - SP$ the class label +1;
4.  Assign each example in $U \cup SP$ the class label -1;
5.  $C \leftarrow$NB($P - SP$, $U \cup SP$);  // Produce a NB classifier
6.  Classify each $u \in U \cup SP$ using $C$;
7.  Decide a probability threshold $t$ using $SP$ and $l$;
8.  For each $u \in U$ do
9.      If its probability $p(+|u) < t$ then
10.         $RN \leftarrow RN \cup \{u\}$;
```

Figure 1. Spy technique for extracting *RN* from *U*

Step 2: Learning using the EM algorithm

Given the positive set *P*, the reliable negative set *RN,* and the remaining unlabeled set *U–RN*, we run EM using NB as the base learning algorithm.

The naive Bayesian (NB) method is an effective text classification algorithm. There are two different NB models, namely, the multinomial NB and the multi-variate Bernoulli NB. In this paper, we use the multinomial NB since it has been observed to perform consistently better than the multi-variate Bernoulli NB (Nigam et al., 2000).

```
Each document in $P$ is assigned the class label 1;
Each document in $RN$ is assigned the class label −1;
Learn an initial NB classifier $f$ from $P$ and $RN$, using Equations (1) and (2);
Repeat
    For each document $d_i$ in $U$-$RN$ do     // E-Step
        Using the current classifier $f$ compute
            $\Pr(c_j|d_i)$ using Equation (3);
    Learn a new NB classifier $f$ from $P$, $RN$ and $U$-$RN$ by computing $\Pr(c_j)$ and $\Pr(w_t|c_j)$, using
    Equations (1) and (2);                     // M-Step
Until the classifier parameters stabilize
The last iteration of EM gives the final classifier $f$;
For each document $d_i$ in $U$ do
    If its probability $\Pr(+|d_i) \geq 0.5$ then
        Output $d_i$ as a positive document;
    else Output $d_i$ as a negative document
```

Figure 2. EM algorithm with the NB classifier

Given a set of training documents D, each document $d_i \in D$ is an ordered list of words. We use $w_{d_i,k}$ to denote the word in position k of d_i, where each word is from the vocabulary $V = \{w_1, \ldots, w_{|v|}\}$, which is the set of all words considered in classification. We also have a set of classes $C = \{c_1, c_2\}$ representing positive and negative classes. For classification, we compute the posterior probability $\Pr(c_j|d_i)$. Based on the Bayes rule and multinomial model, we have

$$\Pr(c_j) = \frac{\sum_{i=1}^{|D|} \Pr(c_j \mid d_i)}{|D|} \quad (1)$$

and with Laplacian smoothing,

$$\Pr(w_t \mid c_j) = \frac{1 + \sum_{i=1}^{|D|} N(w_t, d_i) \Pr(c_j \mid d_i)}{|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N(w_s, d_i) \Pr(c_j \mid d_i)} \quad (2)$$

where $N(w_t,d_i)$ is the number of times that the word w_t occurs in document d_i, and $\Pr(c_j|d_i) \in \{0,1\}$ depending on the class label of the document. Assuming that probabilities of words are independent given the class, we have the NB classifier:

$$\Pr(c_j \mid d_i) = \frac{\Pr(c_j) \prod_{k=1}^{|d_i|} \Pr(w_{d_i,k} \mid c_j)}{\sum_{r=1}^{|C|} \Pr(c_r) \prod_{k=1}^{|d_i|} \Pr(w_{d_i,k} \mid c_r)} \quad (3)$$

EM (Dempster et al. 1977) is a popular class of iterative algorithms for maximum likelihood estimation in problems with incomplete data. It is often used to address missing values in the data by computing expected values using the existing values. The EM algorithm consists of two steps, the E-step and the M-step. The E-step fills in the missing data, and M-step re-estimated the parameters. This process is iterated till satisfaction (i.e. convergence). For NB, the steps used by EM are identical to those used to build the classifier (equations (3) for the E-step, and equations (1) and (2) for the

M-step). In EM, $\Pr(c_j|d_i)$ takes the value in [0, 1] instead of {0, 1} in all the three equations.

The algorithm for the second step of S-EM is given in Figure 2. Lines 1-3 build a NB classifier *f* using *P* and *RN*. Lines 4-8 run EM until convergence. Finally, the converged classifier is used to classify the unlabeled set *U* (lines 10-13).

3.2 Proposed CR-SVM

As we will see in the experiment section, the performance of S-EM can be weak in some cases. This is due to the mixture model assumption of its NB classifier (Nigam et al. 2000), which requires that the mixture components and classes be of one-to-one correspondence. Intuitively, this means that each class should come from a distinctive distribution rather than a mixture of multiple distributions. In our setting, however, the negative class often has documents of mixed topics, e.g., representing the broad class of everything else except the topic(s) represented by the positive class.

There are some existing PU learning methods based on SVM which can deal with this problem, e.g., Roc-SVM (Li and Liu, 2003). Like S-EM, Roc-SVM also has two steps. The first step uses Rocchio classification (Rocchio, 1971) to find a set of reliable negatives *RN* from *U*. In particular, this method treats the entire unlabeled set *U* as negative documents and then uses the positive set *P* and the unlabeled set *U* as the training data to build a Rocchio classifier. The classifier is subsequently applied to classify the unlabeled set *U*. Those documents that are classified as negative are then considered as reliable negative examples *RN*. The second step of Roc-SVM runs SVM iteratively (instead of EM). Unlike NB, SVM does not make any distributional assumption.

However, Roc-SVM does not do well due to the weakness of its first step in finding a good set of reliable negatives *RN*. This motivates us to propose a new SVM based method CR-SVM to detect a better quality *RN* set. The second step of CR-SVM is similar to that in Roc-SVM.

Step 1: Extracting reliable negatives *RN* from *U* using Cosine and Rocchio

The first step of the proposed CR-SVM algorithm for finding a *RN* set consists of two sub-steps:

Sub-step 1 (extracting the potential negative set *PN* using the *cosine similarity*): Given the positive set *P* and the unlabeled set *U*, we extract a set of potential negatives *PN* from *U* by computing the similarities of the unlabeled documents in *U* and the positive documents in *P*. The idea is that those documents in *U* that are very dissimilar to the documents in *P* are likely to be negative.

```
1.  $PN = \emptyset$;
2.  Represent each document in $P$ and $U$ as vectors using the TF-IDF representation;
3.  For each $\mathbf{d}_j \in P$ do
4.      $\mathbf{pr} = \frac{1}{|P|} * \sum_{j=1}^{|P|} \frac{\mathbf{d}_j}{\| \mathbf{d}_j \|_2}$
5.  $\mathbf{pr} = \mathbf{pr} / \| \mathbf{pr} \|_2$;
6.  For each $\mathbf{d}_j \in P$ do
7.      compute $cos(\mathbf{pr}, \mathbf{d}_j)$ using Equation (4);
8.  Sort all the documents $\mathbf{d}_j \in P$ according to $cos(\mathbf{pr}, \mathbf{d}_j)$ in decreasing order;
9.  $\omega = cos(\mathbf{pr}, \mathbf{d}_p)$ where $\mathbf{d}_p$ is ranked in the position of $(1- l)*|P|$;
10. For each $\mathbf{d}_i \in U$ do
11.     If $cos(\mathbf{pr}, \mathbf{d}_i) < \omega$ then
12.         $PN = PN \cup \{\mathbf{d}_i\}$
```

Figure 3. Extracting potential negatives *PN* from *U*

The detailed algorithm is given in Figure 3. Each document in *P* and *U* is first represented as a vector $\mathbf{d} = (q_1, q_2, \ldots, q_n)$ using the TF-IDF scheme (Salton 1986). Each element q_i (i=1, 2, …, n) in **d** represents a word feature w_i. A positive representative vector (**pr**) is built by summing up the documents in *P* and normalizing it (lines 3-5). Lines 6-7 compute the similarities of each document $\mathbf{d}_j$ in *P* with **pr** using the cosine similarity, $cos(\mathbf{pr}, \mathbf{d}_j)$.

Line 8 sorts the documents in *P* according to their $cos(\mathbf{pr}, \mathbf{d}_j)$ values. We want to filter away as many as possible hidden positive documents from *U* so that we can obtain a very pure negative set. Since the hidden positives in *U* should have the same behaviors as the positives in *P* in terms of their similarities to **pr**, we set their minimum similarity as the threshold value ω which is the minimum similarity before a document is considered as a potential negative document:

$$\omega = \min_{j=1}^{|P|} \cos(\mathbf{pr}, \mathbf{d}_j), \mathbf{d}_j \in P \qquad (4)$$

In a noiseless scenario, using the minimum similarity is acceptable. However, most real-life applications contain outliers and noisy artifacts. Using the absolute minimum similarity may be unreliable; the similarity $cos(\mathbf{pr}, \mathbf{d}_j)$ of an outlier docu-

```
1. $RN = \varnothing$;
2. Represent each document in $P$, $PN$ and $U$ as vectors
   using the TF-IDF representation;
3. $\mathbf{p} = \alpha \frac{1}{|P|} \sum_{\mathbf{d}_j \in P} \frac{\mathbf{d}_j}{\|\mathbf{d}_j\|} - \beta \frac{1}{|PN|} \sum_{\mathbf{d}_i \in PN} \frac{\mathbf{d}_i}{\|\mathbf{d}_i\|}$;
4. $\mathbf{n} = \alpha \frac{1}{|PN|} \sum_{\mathbf{d}_i \in PN} \frac{\mathbf{d}_i}{\|\mathbf{d}_i\|} - \beta \frac{1}{|P|} \sum_{\mathbf{d}_j \in P} \frac{\mathbf{d}_j}{\|\mathbf{d}_j\|}$;
5. For each $\mathbf{d}_i \in U$ do
6.     If $cos(\mathbf{d}_i, \mathbf{n}) > cos(\mathbf{d}_i, \mathbf{p})$ then
7.         $RN = RN \cup \{\mathbf{d}_i\}$
```

Figure 4. Identifying *RN* using the Rocchio classifier

ment $\mathbf{d}_j$ in *P* could be near 0 or smaller than most (or even all) negative documents. It would therefore be prudent to ignore a small percentage *l* of the documents in *P* most dissimilar to the representative positive (**pr**) and assume them as noise or outliers. Since we do not know the noise level of the data, to be safe, we use a noise level $l = 5\%$ as the default. The final classification result is not sensitive to *l* as long as it is not too big. In line 9, we use the noise level *l* to decide on a suitable ω. Then, for each document d_i in *U*, if its cosine similarity $cos(\mathbf{pr}, \mathbf{d}_i) < \omega$, we regard it as a potential negative and store it in *PN* (lines 10-12).

Our experiment results showed that *PN* is still not sufficient or big enough for accurate PU learning. Thus, we need to do a bit more work to find the final *RN*.

Sub-step 2 (extracting the final reliable negative set *RN* from *U* using Rocchio with *PN*): At this point, we have a positive set *P* and a potential negative set *PN* where *PN* is a purer negative set than *U*. To extract the final reliable negatives, we employ the Rocchio classification to build a classifier *RC* using *P* and *PN* (We do not use SVM here as it is very sensitive to the noise in *PN*). Those documents in *U* that are classified as negatives by *RC* will then be regarded as reliable negatives, and stored in set *RN*.

The algorithm for this sub-step is given in Figure 4. Following the Rocchio formula, a positive and a negative prototype vectors **p** and **n** are built (lines 3 and 4), which are used to classify the documents in *U* (lines 5-7). α and β are parameters for adjusting the relative impact of the positive and negative examples. In this work, we use $\alpha = 16$ and $\beta = 4$ as recommended in (Buckley et al. 1994).

Step 2: Learning by running SVM iteratively

This step is similar to that in Roc-SVM, building the final classifier by running SVM iteratively with the sets *P*, *RN* and the remaining unlabeled set *Q* ($Q = U - RN$).

The algorithm is given in Figure 5. We run SVM classifiers S_i (line 3) iteratively to extract more and more negative documents from *Q*. The iteration stops when no more negative documents can be extracted from *Q* (line 5). There is, however, a danger in running SVM iteratively, as SVM is quite sensitive to noise. It is possible that during some iteration, SVM is misled by noisy data to extract many positive documents from *Q* and put them in the negative set *RN*. If this happens, the final SVM classifier will be inferior. As such, we employ a test to decide whether to keep the first SVM classifier or the final one. To do so, we use the final SVM classifier obtained at convergence (called S_{last}, line 9) to classify the positive set *P* to see if many positive documents in *P* are classified as negatives. Roc-SVM chooses 5% as the threshold, so CR-SVM also uses this threshold. If there are 5% of positive documents (5%*|*P*|) in *P* that are classified as negative, it indicates that SVM has gone wrong and we should use the first SVM classifier (S_1). In our experience, the first classifier is always quite strong; good results can therefore be achieved even without catching the last (possibly better) classifier.

The main difference between Roc-SVM and CR-SVM is that Roc-SVM does not produce *PN*. It simply treats the unlabeled set *U* as negatives for extracting *RN*. Since *PN* is clearly a purer negative set than *U*, the use of *PN* by CR-SVM helps extract a better quality reliable negative set *RN* which subsequently allows the final classifier of CR-SVM to give better results than Roc-SVM.

Note that the methods (S-EM and CR-SVM) are all two-step algorithms in which the first step and the second step are independent of each other. The algorithm for the second step basically needs a good set of reliable negatives *RN* extracted from *U*. This means that one can pick any algorithm for the first step to work with any algorithm for the second step. For example, we can also have CR-EM which uses the algorithm (shown in Figures 3 and 4) of the first step of CR-SVM to combine with the algorithm of the second step of S-EM. CR-EM actually works quite well as it is also able to exploit the more accurate reliable negative set *RN* extracted using cosine and Rocchio.

4 Empirical Evaluation

We now present the experimental results to support our claim that negative training data is not needed and can even harm text classification. We also show the effectiveness of the proposed PU learning methods CR-SVM and CR-EM. The following methods are compared: (1) traditional supervised learning methods SVM and NB which use both positive and negative training data; (2) PU learning methods, including two existing methods S-EM and Roc-SVM and two new methods CR-SVM and CR-EM, and (3) one-class SVM (Schölkop et al., 1999) where only positive training data is used in learning (the unlabeled set is not used at all).

We used LIBSVM[1] for SVM and one-class SVM, and two publicly available[2] PU learning techniques S-EM and Roc-SVM. Note that we do not compare with some other PU learning methods such as those in (Liu et al. 2003, Lee and Liu, 2003 and Elkan and Noto, 2008) as the purpose of this paper is not to find the best PU learning method but to show that PU learning can address our special sample selection bias problem. Our current methods already do very well for this purpose.

4.1 Datasets and Experimental Settings

We used two well-known benchmark data collections for text classification, the Reuters-21578 collection[3] and the 20 Newsgroup collection[4]. Reuters-21578 contains 21578 documents. We used the most populous 10 out of the 135 categories following the common practice of other researchers. 20 Newsgroup has 11997 documents from 20 discussion groups. The 20 groups were also categorized into 4 main categories.

We have performed two sets of experiments, and just used bag-of-words as features since our objective in this paper is not feature engineering.

(1) **Test set has *other topic* documents.** This set of experiments simulates the scenario in which the negative training and test samples have different distributions. We select positive, negative and *other topic* documents for Reuters and 20 Newsgroup, and produce various data sets. Using these data sets, we want to show that PU learning can do better than traditional learning that uses both positive and negative training data.

```
1.  Every document in $P$ is assigned the class label +1;
2.  Every document in $RN$ is assigned the label –1;
3.  Use $P$ and $RN$ to train a SVM classifier $S_i$, with $i$ =
    1 initially and $i = i+1$ with each iteration (line 3-7);
4.  Classify $Q$ using $S_i$. Let the set of documents in $Q$
    that are classified as negative be $W$;
5.  If ($W = \emptyset$) then stop;
6.  else $Q = Q – W$;
7.          $RN = RN \cup W$
8.  goto (3);
9.  Use the last SVM classifier $S_{last}$ to classify $P$;
10. If more than 5% positives are classified as negative
11.     then use $S_1$ as the final classifier;
12. else use $S_{last}$ as the final classifier;
```

Figure 5. Constructing the final classifier using SVM

For the Reuters collection, each of the 10 categories is used as a positive class. We randomly select one or two of the remaining categories as the negative class (denoted by Neg 1 or Neg 2), and then we randomly choose some documents from the rest of the categories as *other topic* documents. These *other topic* documents are regarded as negatives and added to the test set but not to the negative training data. They thus introduce a different distribution to the negative test data. We generated 20 data sets (10*2) for our experiments this way.

The 20 Newsgroup collection has 4 main categories with sub-categories[5]; the sub-categories in the same main category are relatively similar to each other. We are able to simulate two scenarios: (1) the *other topic* documents are similar to the negative class documents (*similar case*), and (2) the *other topic* documents are quite different from the negative class documents (*different case*). This allows us to investigate whether the classification results will be affected when the *other topic* documents are somewhat similar or vastly different from the negative training set. To create the training and test data for our experiments, we randomly select one sub-category from a main category (*cat* 1) as the positive class, and one (or two) sub-category from another category (*cat* 2) as the negative class (again denoted by Neg 1 or Neg 2). For the *other topics*, we randomly choose some docu-

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

[2] http://www.cs.uic.edu/~liub/LPU/LPU-download.html

[3] http://www.research.att.com/~lewis/reuters21578.html

[4] http://people.csail.mit.edu/jrennie/20Newsgroups/

[5] The four main categories and their corresponding sub-categories are: *computer* (graphics, os, ibmpc.hardware, mac.hardware, windows.x), *recreation* (autos, motorcycles, baseball, hockey), *science* (crypt, electronics, med, space), and *talk* (politics.misc, politics.guns, politics.mideast, religion).

ments from the remaining sub-categories of *cat* 2 for the *similar case*, and some documents from a randomly chosen different category (*cat* 3) (as the *other topic* documents) for the *different case*. We generated 8 data sets (4*2) for the *similar case*, and 8 data sets (4*2) for the *different case*.

The training and test sets are then constructed as follows: we partition the positive (and similarly for the negative) class documents into two standard subsets: 70% for training and 30% for testing. In order to create different experimental settings, we vary the number of the *other topic* documents that are added to the test set as negatives, controlled by a parameter α, which is a percentage of $|TN|$, where $|TN|$ is the size of the negative test set without the *other topic* documents. That is, the number of *other topic* documents added is $\alpha \times |TN|$.

(2) **Test set has no *other topic* documents.** This set of experiments is the traditional classification in which the training and test data have the same distribution. We employ the same data sets as in (1) but without having any *other topic* documents in the test set. Here we want to show that PU learning can do equally well without using the negative training data even in the traditional setting.

4.2 Results with *Other Topic* Documents in Test Set

We show the results for experiment set (1), i.e. the distributions of the negative training and test data are different (caused by the inclusion of *other topic* documents in the test set, or the addition of *other topic* documents to complement existing negatives in the test set). The evaluation metric is the F-score on the positive class (Bollmann and Cherniavsky, 1981), which is commonly used for evaluating text classification.

4.2.1 Results on the Reuters data

Figure 6 shows the comparison results when the negative class contains only one category of documents (Neg 1), while Figure 7 shows the results when the negative class contains documents from two categories (Neg 2) in the Reuters collection. The data points in the figures are the averages of the results from the corresponding datasets.

Our proposed method CR-SVM is shown to perform consistently better than the other techniques. When the size of the *other topic* documents (*x*-axis) in the test set increases, the F-scores of the two traditional learning methods SVM and NB decreased much more dramatically as compared with the PU learning techniques. The traditional learning models were clearly unable to handle different distributions for training and test data. Among the PU learning techniques, the proposed CR-SVM gave the best results consistently. Roc-SVM did not do consistently well as it did not manage to find high quality reliable negatives *RN* sometimes. The EM based methods (CR-EM and S-EM) performed well in the case when we had only one negative class (Figure 6). However, it did not do well in the situation where there were two negative classes (Figure 7) due to the underlying mixture model assumption of the naïve Bayesian classifier. One-class SVM (OSVM) performed poorly because it did not exploit the useful information in the unlabeled set at all.

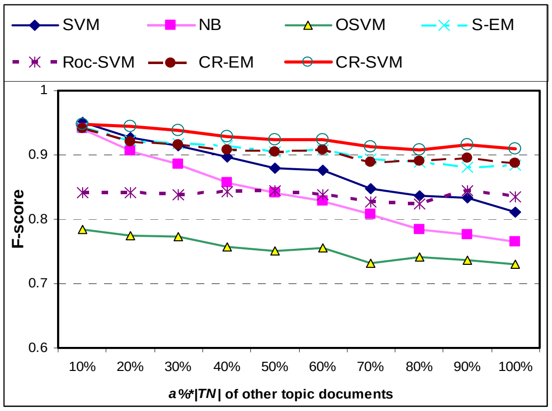


Figure 6. Results of Neg 1 using the Reuter data

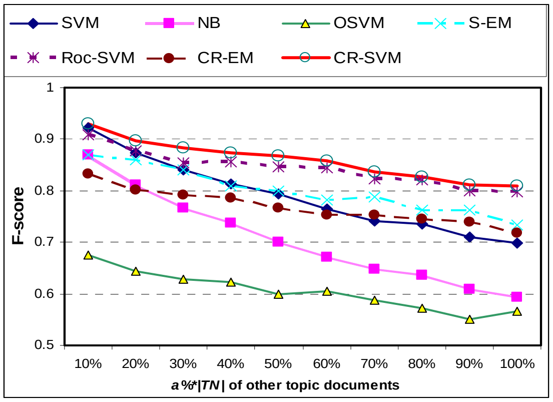


Figure 7. Results of Neg 2 using the Reuter data

4.2.2 Results on 20 Newsgroup data

Recall that for the 20 Newsgroup data, we have two settings: *similar case* and *different case*.

Similar case: Here, the *other topic* documents are similar to the *negative* class documents, as they belong to the same main category.

The comparison results are given in Figure 8 (Neg 1) and Figure 9 (Neg 2). We observe that CR-EM, S-EM and CR-SVM all performed well. EM based methods (CR-EM and S-EM) have a slight edge over CR-SVM. Again, the F-scores of the traditional supervised learning (SVM and NB) deteriorated when more *other topic* documents were added to the test set, while CR-EM, S-EM and CR-SVM were able to remain unaffected and maintained roughly constant F-scores. When the negative class contained documents from two categories (Neg 2), the F-scores of the traditional learning dropped even more rapidly. Both Roc-SVM and One-class SVM (OSVM) performed poorly, due to the same reasons given previously.

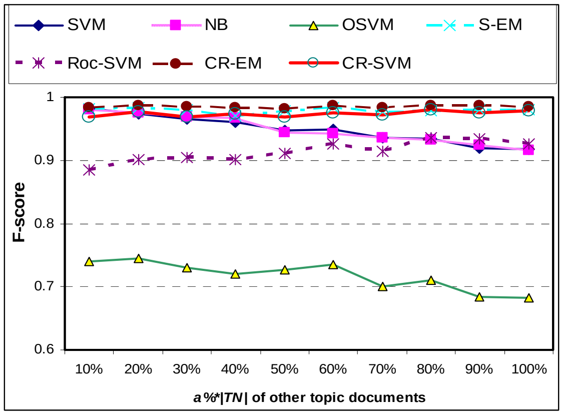


Figure 8. Results of Neg 1, *similar case* – using the 20 Newsgroup data

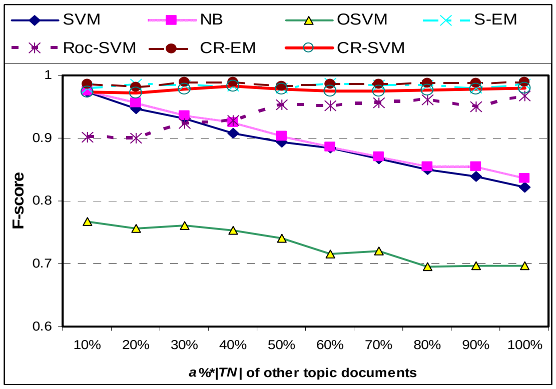


Figure 9. Results of Neg 2, *similar case* – using the 20 Newsgroup data

Different case: In this case, the *other topic* documents are quite different from the negative class documents, since they are originated from different main categories.

The results are shown in Figures 10 (Neg 1) and 11 (Neg 2). The trends are similar to those for the *similar case*, except that the performance of the traditional supervised learning methods (SVM and NB) dropped even more rapidly with more *other topic* documents. As the *other topic* documents have very different distributions from the negatives in the training set in this case, they really confused the traditional classifiers. In contrast, the three PU learning techniques were still able to perform consistently well, regardless of the number of *other topic* documents added to the test data.

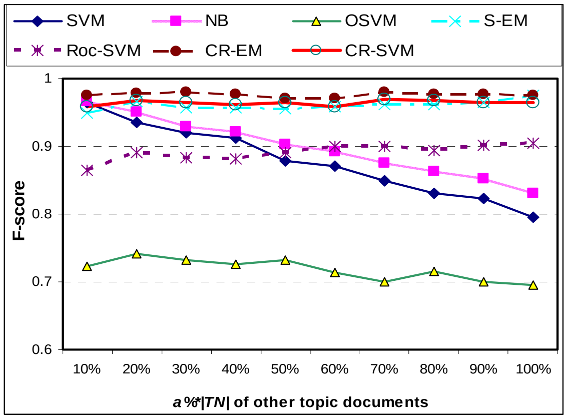


Figure 10. Results of Neg 1, *different case* – using the 20 Newsgroup data

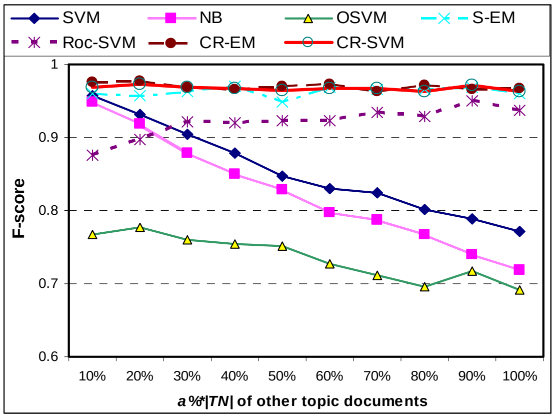


Figure 11. Results of Neg 2, *different case* – using the 20 Newsgroup data

In summary, the results showed that learning with negative training data based on the traditional paradigm actually harms classification when the identical distribution assumption does not hold.

4.3 Results without *Other Topic* Documents in Test Set

Given an application, one may not know whether the identical distribution assumption holds. The above results showed that PU learning is better when it does not hold. How about when the assumption does hold? To find out, we compared the results of SVM, NB, and three PU learning methods using the datasets without any *other topic* documents added to the test set. In this case, the training and test data distributions are the same.

Table 1 shows the results for this scenario. Note that for PU learning, the negative training data were not used. The traditional supervised learning techniques (SVM and NB), which made full use of the positive and negative training data, only performed just about 1-2% better than the PU learning method CR-SVM (which is not statistically significant based on paired *t*-test). This suggests that we can do away with negative training data, since PU learning can perform equally well without them. This has practical importance since the full coverage of negative training data is hard to find and to label in many applications.

From the results in Figures 6–11 and Table 1, we can conclude that PU learning can be used for binary text classification without the negative training data (which can be harmful for the task). CR-SVM is our recommended PU learning method based on its generally consistent performance.

Table 1. Comparison of methods without *other* documents in test set

Methods	Reuters (Neg 1)	Reuters (Neg 2)	20News (Neg 1)	20News (Neg 2)
SVM	0.971	0.964	0.988	0.990
NB	0.972	0.947	0.988	0.992
S-EM	0.952	0.921	0.974	0.975
CR-EM	0.955	0.897	0.983	0.986
CR-SVM	0.960	0.959	0.967	0.974

5 Conclusions

This paper studied a special case of the sample selection bias problem in which the positive training and test distributions are the same, but the negative training and test distributions may be different. We showed that in this case, the negative training data should not be used in learning, and PU learning can be applied to this setting. A new PU learning algorithm (called CR-SVM) was also proposed to overcome the weaknesses of the current two-step algorithms.

Our experiments showed that the traditional classification methods suffered greatly when the distributions are different for the negative training and test data, but PU learning does not. We also showed that PU learning performed equally well in the ideal case where the training and test data have identical distributions. As such, it can be advantageous to discard the potentially harmful negative training data and use PU learning for classification.

In our future work, we plan to do more comprehensive experiments to compare the classic supervised learning and PU learning techniques with different kinds of settings, for example, by varying the ratio between positive and negative examples, as well as their sizes. It is also important to explore how to catch the best iteration of the SVM/NB classifier in the iterative running process of the algorithms. Finally, we would like to point out that it is conceivable that negative training data could still be useful in many cases. An interesting direction to explore is to somehow combine the extracted reliable negative data from the unlabeled set and the existing negative training data to further enhance learning algorithms.

References

Agirre E., Lacalle L.O. 2009. *Supervised Domain Adaption for WSD*. Proceedings of the 12th Conference of the European Chapter for Computational Linguistics (EACL09), pp 42-50.

Andrew A., Nallapati R., Cohen W., 2008. *Exploiting Feature Hierarchy for Transfer Learning in Named Entity Recognition*, ACL.

Bickel, S., Bruckner, M., and Scheffer. 2009. T. *Discriminative learning under covariate shift*. Journal of Machine Learning Research.

Bickel S. and Scheffer T. 2007. *Dirichlet-enhanced spam filtering based on biased samples*. In Advances in Neural Information Processing Systems.

Bollmann, P.,& Cherniavsky, V. 1981. *Measurement-theoretical investigation of the mz-metric*. Information Retrieval Research.

Buckley, C., Salton, G., & Allan, J. 1994. *The effect of adding relevance information in a relevance feedback environment*, SIGIR.

Blum, A. and Mitchell, T. 1998. *Combining labeled and unlabeled data with co-training*. In Proc. of Computational Learning Theory, pp. 92–10.

Chan Y. S., Ng H. T. 2007. *Domain Adaptation with Active Learning for Word Sense Disambiguation*, ACL.

Dempster A., Laird N. and Rubin D.. 1977. *Maximum likelihood from incomplete data via the EM algorithm*, Journal of the Royal Statistical Society.

Denis F., *PAC learning from positive statistical queries*. ALT, 1998.

Denis F., Laurent A., Rémi G., Marc T. 2003. *Text classification and co-training from positive and unlabeled examples*. ICML.

Denis, F, Rémi G, and Marc T. 2002. *Text Classification from Positive and Unlabeled Examples.* In Proceedings of the 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems.

Downey, D., Broadhead, M. and Etzioni, O. 2007. *Locating complex named entities in Web Text.* IJCAI.

Dudik M., Schapire R., and Phillips S. 2005. *Correcting sample selection bias in maximum entropy density estimation*. In Advances in Neural Information Processing Systems.

Elkan, C. and Noto, K. 2008. *Learning classifiers from only positive and unlabeled data*. KDD, 213-220.

Goldwasser, D., Roth D. 2008. *Active Sample Selection for Named Entity Transliteration*, ACL.

Heckman J. 1979. *Sample selection bias as a specification error*. Econometrica, 47:153–161.

Huang J., Smola A., Gretton A., Borgwardt K., and Scholkopf B. 2007. *Correcting sample selection bias by unlabeled data*. In Advances in Neural Information Processing Systems.

Jiang J. and Zhai C. X. 2007. *Instance Weighting for Domain Adaptation in NLP*, ACL.

Lee, W. S. and Liu, B. 2003. *Learning with Positive and Unlabeled Examples Using Weighted Logistic Regression*. ICML.

Lewis D. 1995. *A sequential algorithm for training text classifiers*: *corrigendum and additional data*. SIGIR Forum, 13-19.

Li, S., Zong C., 2008. *Multi-Domain Sentiment Classification*, ACL.

Li, X., Liu, B. 2003. *Learning to classify texts using positive and unlabeled data*, IJCAI.

Li, X., Liu, B., 2005. *Learning from Positive and Unlabeled Examples with Different Data Distributions*. ECML.

Li, X., Liu, B., 2007. *Learning to Identify Unexpected Instances in the Test Set*. IJCAI.

Li, X., Yu, P. S., Liu B., and Ng, S. 2009. *Positive Unlabeled Learning for Data Stream Classification*, SDM.

Li, X., Zhang L., Liu B., and Ng, S. 2010. *Distributional Similarity vs. PU Learning for Entity Set Expansion*, ACL.

Liu, B, Dai, Y., Li, X., Lee, W-S., and Yu. P. 2003. *Building text classifiers using positive and unlabeled examples*. ICDM, 179-188.

Liu, B, Lee, W-S, Yu, P. S, and Li, X. 2002. *Partially supervised text classification.* ICML, 387-394.

Nigam, K., McCallum, A., Thrun, S. and Mitchell, T. 2000. *Text classification from labeled and unlabeled documents using EM*. Machine Learning, 39(2/3), 103–134.

Pan, S. J. and Yang, Q. 2009. *A survey on transfer learning*. IEEE Transactions on Knowledge and Data Engineering, Vol. 99, No. 1.

Rocchio, J. 1971. Relevant feedback in information retrieval. In G. Salton (ed.). *The smart retrieval system: experiments in automatic document processing*,

Englewood Cliffs, NJ, 1971.Sagae K., Tsujii J. 2008. *Online Methods for Multi-Domain Learning and Adaptation*, EMNLP.

Salton G. and McGill M. J. 1986. *Introduction to Modern Information Retrieval.*

Schölkop f B., Platt J.C., Shawe-Taylor J., Smola A.J., and Williamson R.C. 1999. *Estimating the support of a high-dimensional distribution*. Technical report, Microsoft Research, MSR-TR-99-87.

Shimodaira H. 2000. *Improving predictive inference under covariate shift by weighting the log-likelihood function*. Journal of Statistical Planning and Inference, 90:227–244.

Sugiyama M. and Muller K.-R. 2005. *Input-dependent estimation of generalization error under covariate shift*. Statistics and Decision, 23(4):249–279.

Sugiyama M., Nakajima S., Kashima H., von Bunau P., and Kawanabe M. 2008. *Direct importance estimation with model selection and its application to covariate shift adaptation*. In Advances in Neural Information Processing Systems.

Tsuboi J., Kashima H., Hido S., Bickel S., and Sugiyama M. 2008. *Direct density ratio estimation for large-scale covariate shift adaptation*. In Proceedings of the SIAM International Conference on Data Mining, 2008.

Wu D., Lee W.S., Ye N. and Chieu H. L. 2009. *Domain adaptive bootstrapping for named entity recognition*, ACL.

Wu Q., Tan S. and Cheng X. 2009. *Graph Ranking for Sentiment Transfer*, ACL.

Yang Q., Chen Y., Xue G., Dai W., Yu Y. 2009. *Heterogeneous Transfer Learning for Image Clustering via the SocialWeb*, ACL

Yu, H., Han, J., K. Chang. 2002. *PEBL: Positive example based learning for Web page classification using SVM*. KDD, 239-248.

Zadrozny B. 2004. *Learning and evaluating classifiers under s ample selection bias*, ICML.

Zhou Z., Gao J., Soong F., Meng H. 2006. *A Comparative Study of Discriminative Methods for Reranking LVCSR N-best Hypotheses in Domain Adaptation and Generalization*. ICASSP.

Modeling Organization in Student Essays

Isaac Persing and **Alan Davis** and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{persingq,alan,vince}@hlt.utdallas.edu

Abstract

Automated essay scoring is one of the most important educational applications of natural language processing. Recently, researchers have begun exploring methods of scoring essays with respect to particular dimensions of quality such as coherence, technical errors, and relevance to prompt, but there is relatively little work on modeling organization. We present a new annotated corpus and propose heuristic-based and learning-based approaches to scoring essays along the organization dimension, utilizing techniques that involve sequence alignment, alignment kernels, and string kernels.

1 Introduction

Automated essay scoring, the task of employing computer technology to evaluate and score written text, is one of the most important educational applications of natural language processing (NLP) (see Shermis and Burstein (2003) and Shermis et al. (2010) for an overview of the state of the art in this task). Recent years have seen a surge of interest in this and other educational applications in the NLP community, as evidenced by the panel discussion on "Emerging Application Areas in Computational Linguistics" at NAACL 2009, as well as increased participation in the series of workshops on "Innovative Use of NLP for Building Educational Applications". Besides its potential commercial value, automated essay scoring brings about a number of relatively less-studied but arguably rather challenging discourse-level problems that involve the computational modeling of different facets of text structure, such as content, coherence, and organization.

A major weakness of many existing essay scoring engines such as IntelliMetric (Elliot, 2001) and Intelligent Essay Assessor (Landauer et al., 2003) is that they adopt a holistic scoring scheme, which summarizes the quality of an essay with a single score and thus provides very limited feedback to the writer. In particular, it is not clear which dimension of an essay (e.g., coherence, relevance) a score should be attributed to. Recent work addresses this problem by scoring a particular dimension of essay quality such as coherence (Miltsakaki and Kukich, 2004), technical errors, and relevance to prompt (Higgins et al., 2004). Automated systems that provide instructional feedback along multiple dimensions of essay quality such as *Criterion* (Burstein et al., 2004) have also begun to emerge.

Nevertheless, there is an essay scoring dimension for which few computational models have been developed — *organization*. Organization refers to the structure of an essay. A high score on organization means that writers introduce a topic, state their position on that topic, support their position, and conclude, often by restating their position (Silva, 1993). A well-organized essay is structured in a way that logically develops an argument. Note that organization is a different facet of text structure than coherence, which is concerned with the transition of ideas at both the global (e.g., paragraph) and local (e.g., sentence) levels. While organization is an important dimension of essay quality, state-of-the-art essay scoring software such as e-rater V.2 (Attali and Burstein, 2006) employs rather simple heuristic-based methods for computing the score of an essay along this particular dimension.

Our goal in this paper is to develop a computational model for the organization of student es-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 229–239,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

says. While many models of text coherence have been developed in recent years (e.g., Barzilay and Lee (2004), Barzilay and Lapata (2005), Soricut and Marcu (2006), Elsner et al. (2007)), the same is not true for text organization. One reason is the availability of training and test data for coherence modeling. Coherence models are typically evaluated on the sentence ordering task, and hence training and test data can be generated simply by scrambling the order of the sentences in a text. On the other hand, it is not particularly easy to find poorly organized texts for training and evaluating organization models. We believe that student essays are an ideal source of well- and poorly-organized texts. We evaluate our organization model on a data set of 1003 essays annotated with organization scores.

In sum, our contributions in this paper are two-fold. First, we address a less-studied discourse-level task — predicting the organization score of an essay — by developing a computational model of organization, thus establishing a baseline against which future work on this task can be compared. Second, we annotate a subset of our student essay corpus with organization scores and make this data set publicly available. Since progress in organization modeling is hindered in part by the lack of a publicly annotated corpus, we believe that our data set will be a valuable resource to the NLP community.

2 Corpus Information

We use as our corpus the 4.5 million word International Corpus of Learner English (ICLE) (Granger et al., 2009), which consists of more than 6000 essays written by university undergraduates from 16 countries and 16 native languages who are learners of English as a Foreign Language. 91% of the ICLE texts are argumentative. The essays we used vary greatly in length, containing an average of 31.1 sentences in 7.5 paragraphs, averaging 4.1 sentences per paragraph. About one quarter of the essays had five or fewer paragraphs, and another quarter contained nine or more paragraphs. Similarly, about one quarter of essays contained 24 or fewer sentences and the longest quarter contained 36 or more sentences

We selected a subset consisting of 1003 essays from the ICLE to annotate and use for training and testing of our model of essay organization. While narrative writing asks students to compose descriptive stories, *argumentative* (also known as *persuasive*) writing requires students to state their opinion on a topic and to validate that opinion with convincing arguments. For this reason, we selected only argumentative essays rather than narrative pieces, because they contain the discourse structures and kind of organization we are interested in modeling.

Topic	Languages	Essays
Most university degrees are theoretical and do not prepare students for the real world. They are therefore of very little value.	13	147
The prison system is outdated. No civilized society should punish its criminals: it should rehabilitate them.	11	103
In his novel *Animal Farm*, George Orwell wrote “All men are equal but some are more equal than others.” How true is this today?	10	82

Table 1: Some examples of writing topics.

To ensure representation across native languages of the authors, we selected mostly essays written in response to topics which are well-represented in multiple languages. This avoids many issues that may arise when certain vocabulary is used in response to a particular topic for which essays written by authors from only a few languages are available. Table 1 shows three of the twelve topics selected for annotation. Fifteen native languages are represented in the set of essays selected for annotation.

3 Corpus Annotation

To develop our essay organization model, human annotators scored 1003 essays using guidelines in an essay annotation rubric. Annotators evaluated the organization of each essay using a numerical score from 1 to 4 at half-point increments. This contrasts with previous work on essay scoring, where the corpus is annotated with a binary decision (i.e., *good* or *bad*) for a given scoring dimension (e.g., Higgins et al. (2004)). Hence, our annotation scheme not only provides a finer-grained distinction of organization quality (which can be important in practice), but also makes the prediction task more challenging.

The meaning of each integer score was described and discussed in detail. Table 2 shows the description of each score for the organization dimension.

Score	Description of Essay Organization
4	essay is **well structured** and is organized in a way that logically develops an argument
3	essay is **fairly well structured** but could somewhat benefit from reorganization
2	essay is **poorly structured** and would greatly benefit from reorganization
1	essay is **completely unstructured** and requires major reorganization

Table 2: Descriptions of the meaning of each score.

Our annotators were selected from over 30 applicants who were familiarized with the scoring rubric and given sample essays to score. The six who were most consistent with the expected scores were given additional essays to annotate. To ensure consistency in scoring, we randomly selected a large subset of our corpus (846 essays) to have graded by two different annotators. Analysis of these doubly annotated essays reveals that, though annotators only exactly agree on the organization score of an essay 29% of the time, the scores they apply are within 0.5 points in 71% of essays and within 1.0 point in 93% of essays. Additionally, if we treat one annotator's scores as a gold standard and the other annotator's scores as predictions, the predicted scores have a mean error of 0.54 and a mean squared error of 0.50. Table 3 shows the number of essays that received each of the seven scores for organization.

score	1.0	1.5	2.0	2.5	3.0	3.5	4.0
essays	24	14	35	146	416	289	79

Table 3: Distribution of organization scores.

4 Function Labeling

As mentioned before, a high score on organization means that writers introduce a topic, support their position, and conclude. If one or more of these elements are missing or if they appear out of order (e.g., the conclusion appears before the introduction), the resulting essay will typically be considered poorly organized. Hence, knowing the *discourse function label* of each paragraph in an essay would be helpful for predicting its organization score.

Two questions naturally arise. First, how can we obtain the discourse function label of each paragraph? One way is to automatically acquire such labels from a corpus of student essays where each paragraph is annotated with its discourse function label. To our knowledge, however, there is no publicly available corpus that is annotated with such information. As a result, we will resort to labeling a paragraph with its function label heuristically.

Second, which paragraph function labels would be most useful for scoring the organization of an essay? Based on our linguistic intuition, we identify four potentially useful paragraph function labels: Introduction, Body, Rebuttal, and Conclusion. Table 4 gives the descriptions of these labels.

Label	Name	Paragraph Type
I	Introduction	introduces essay topic and states author's position and main ideas
B	Body	provides reasons, evidence, and examples to support main ideas
C	Conclusion	summarizes and concludes arguments made in body paragraphs
R	Rebuttal	considers counter-arguments to thesis or main ideas

Table 4: Descriptions of paragraph function labels.

Setting aside for the moment the problem of exactly how to predict an essay's organization score given its paragraph sequence, the problem of obtaining paragraph labels to use for this task still remains. As mentioned above, we adopt a heuristic approach to paragraph function labeling. The question, then, is: what kind of knowledge sources should our heuristics be based on? We have identified two types of knowledge sources that are potentially useful for paragraph labeling. The first of these are positional, dealing with where in the essay a paragraph appears. So for example, the first paragraph in an essay is likely to be an Introduction, while the last is likely to be a Conclusion. A paragraph in any other position, on the other hand, is more likely to be a Body or Rebuttal paragraph.

Label	Name	Sentence Function
P	Prompt	restates the prompt given to the author and contains no new material or opinions
T	Transition	shifts the focus to new topics but contains no meaningful information
H	Thesis	states the author's position on the topic for which he/she is arguing
M	Main Idea	asserts reasons and foundational arguments that support the thesis
E	Elaboration	further explains reasons and ideas but contains no evidence or examples
S	Support	provides evidence and examples to support the claims made in other statements
C	Conclusion	summarizes and concludes the entire argument or one of the main ideas
R	Rebuttal	considers counter-arguments that contrast with the thesis or main ideas
O	Solution	puts to rest the questions and problems brought up by counter-arguments
U	Suggestion	proposes solutions the problems brought up by the argument

Table 5: Descriptions of sentence function labels.

A second potentially useful knowledge source involves the types of sentences appearing in a paragraph. This idea presupposes that, like paragraphs, sentences too can have discourse function labels indicating the logical role they play in an argument. The sentence label schema we propose, which is described in Table 5, is based on work in discourse structure by Burstein et al. (2003), but features additional sentence labels.

To illustrate why these sentence function labels may be useful for paragraph labeling, consider a paragraph containing a Thesis sentence. The presence of a Thesis sentence is a strong indicator that the paragraph containing it is either an Introduction or Conclusion. Similarly, a paragraph containing Rebuttal or Solution sentences is more likely to be a Body or Rebuttal paragraph.

Hence, to obtain a paragraph's function label, we need to first label its sentences. However, we are faced with the same problem: how can we obtain the sentence function labels? One way is to learn them from a corpus where each sentence is manually annotated with its sentence function label, which is the approach adopted by Burstein et al. (2003). However, this annotated corpus is not publicly available. In fact, to our knowledge, there is no publicly-available corpus that is annotated with sentence function labels. Consequently, we adopt a heuristic approach to sentence function labeling.

Overall, we created a knowledge-lean set of heuristic rules labeling paragraphs and sentences. Because many of the paragraph labeling heuristics depend on the availability of sentence labels, we will describe the sentence labeling heuristics first. For each sentence function label x, we identify several features whose presence increases our confidence that a given sentence is an example of x. So for example, the presence of any of the words "agree", "think", or "opinion" increases our confidence that the sentence they occur in is a Thesis. If the sentence instead contains words such as "however", "but", or "argue", these increase our confidence that the sentence is a Rebuttal. The features we examine for sentence labeling are not limited to words, however. Each content word the sentence shares with the essay prompt gives us evidence that the sentence is a restatement of the prompt. Having searched a sentence for all these clues, we finally assign the sentence the function label having the most support among the clues found.

The heuristic rules for paragraph labeling are similar in nature, though they depend heavily on the labels of a paragraph's component sentences. If a paragraph contains Thesis, Prompt, or Background sentences, the paragraph is likely to be an Introduction. However, if a paragraph contains Main Idea, Support, or Conclusion sentences, it is likely to be a Body paragraph. Finally, as mentioned previously, some positional information is used in labeling paragraphs. For example, a paragraph that is the first paragraph in an essay is likely to be an Introduction, but a paragraph that is neither the first nor the last is likely to be either a Rebuttal or Body paragraph. After searching a paragraph for all these features, we gather the pieces of evidence in support of each paragraph label and assign the paragraph the label having the most support.[1]

[1] Space limitations preclude a complete listing of these para-

graph and sentence labeling heuristics. See our website at `http://www.hlt.utdallas.edu/~alan/ICLE/` for the complete list of heuristics.

5 Heuristic-Based Organization Scoring

Having applied labels to each paragraph in an essay, how can we use these labels to predict the essay's score? Recall that the importance of each paragraph label stems not from the label itself, but from the sequence of labels it appears in. Motivated by this observation, we exploit a technique that is commonly used in bioinformatics — *sequence alignment*. While sequence alignment has also been used in text and paraphrase generation (e.g., Barzilay and Lee (2002; 2003)), it has not been extensively applied to other areas of language processing, including essay scoring. In this section, we will present two heuristic approaches to organization scoring, one based on aligning *paragraph sequences* and the other on aligning *sentence sequences*.

5.1 Aligning Paragraph Sequences

As mentioned above, our first approach to heuristic organization scoring involves aligning paragraph sequences. Specifically, this approach operates in two steps. Given an essay e in the test set, we (1) find the k essays in the training set that are most similar to e via paragraph sequence alignment, and then (2) predict the organization score of e by aggregating the scores of its k nearest neighbors obtained in the first step. Below we describe these two steps in detail.

First, to obtain the k nearest neighbors of e, we employ the Needleman-Wunsch alignment algorithm (Needleman and Wunsch, 1970), which computes a similarity score for any pair of essays by finding an optimal alignment between their paragraph sequences. To illustrate why we believe sequence alignment can help us determine which essays are most similar, consider two example essays. One essay, which we will call IBBBC, begins with an Introductory paragraph, follows it with three Body paragraphs, and finally ends with a Concluding paragraph. Another essay CRRRI begins with a paragraph stating its Conclusion, follows it with three Rebuttal paragraphs, and ends with a paragraph Introducing the essay's topic. We can tell by a casual glance at the sequences that any reasonable similarity function should tell us that they are not very similar. The Needleman-Wunsch alignment algorithm has this effect since the score of the alignment it produces would be hurt by the facts that (1) there is not much overlap in the sets of paragraph labels each contains, and (2) the paragraph labels they do share (I and C) do not occur in the same order. The resulting alignment would therefore contain many mismatches or indels.[2]

If we now consider a third essay whose paragraph sequence could be represented as IBRBC, a good similarity function should tell us that IBBBC and IBRBC are very similar. The Needleman-Wunsch alignment score between the two paragraph sequences has this property, as the alignment algorithm would discover that the two sequences are identical except for the third paragraph label, which could be mismatched for a small penalty. We would therefore conclude that the IBBBC and IBRBC essays should receive similar organization scores.

To fully specify how to find the k nearest neighbors of an essay, we need to define a similarity function between paragraph labels. In sequence alignment, similarity function $S(i, j)$ tells us how likely it is that symbol i (in our case, a paragraph label) will be substituted with another symbol j. While we expect that in an alignment between high-scoring essays, an Introduction paragraph is most likely to be aligned with another Introduction paragraph, how much worse should the alignment score be if an Introduction paragraph needs to be mismatched with a Rebuttal paragraph or replaced with an indel? We solve this problem by heuristically defining the similarity function as follows: $S(i, j) = 1$ when $i = j$, $S(i, j) = -1$ when $i \neq j$, and also $S(i, -) = S(-, i) = -1$, where '$-$' is an indel. In other words, the similarity function encourages the alignment between two identical function labels and discourages the alignment between two different function labels, regardless of the type of function labels.

After obtaining the k nearest neighbors of e, the next step is to predict the organization score of e by aggregating the scores of its k nearest neighbors into one number. (Note that we know the organiza-

[2] In pairwise sequence alignment, a mismatch occurs when one symbol has to be substituted for another to make two sequences match. An indel indicates that in order to transform one sequence to match another, we must either **in**sert a symbol into one sequence or **del**ete a symbol from the other sequence.

tion score of each nearest neighbor, since they are all taken from the training set.) One natural way to do this would be to take the mean, median, or mode of its k nearest neighboring essays from the training set. Hence, our first heuristic method H_p for scoring organization has three variants.

5.2 Aligning Sentence Sequences

An essay's paragraph sequence captures information about its organization at a high level, but ignores much of its lower level structure. Since we have also heuristically labeled sentences, it now makes sense to examine the sequences of sentence function labels within an essay's paragraphs. The intuition is that at least some portion of an essay's organization score can be attributed to the organization of the sentence sequences of its component paragraphs.

To address this concern, we propose a second heuristic approach to organization scoring. Given a test essay e, we first find for each *paragraph* in e the k *paragraphs* in the training set that are most similar to it. Specifically, each paragraph is represented by its sequence of *sentence* function labels. Given this paragraph representation, we can find the k nearest neighbors of a paragraph by applying the Needleman-Wunsch algorithm described in the previous subsection to align *sentence* sequences, using the same similarity function we defined above.

Next, we score each paragraph p_i by aggregating the scores of its k nearest neighbors obtained in the first step, assuming the score of a nearest neighbor paragraph is the same as the organization score of the training set essay containing it. As before, we can employ the mean, median, or mode to aggregate the scores of the nearest neighbors of p_i.

Finally, we predict the organization score of e by aggregating the scores of its paragraphs obtained in the second step. Again, we can employ mean, median, or mode to aggregate the scores. Since we have three ways of aggregating the scores of a paragraph's nearest neighbors and three ways of aggregating the resulting paragraph scores, this second method H_s for scoring organization has nine variants.

6 Learning-Based Organization Scoring

In the previous section, we proposed two heuristic approaches to organization scoring, one based on aligning paragraph label sequences and the other based on aligning sentence label sequences. In the process of constructing these two systems, however, we created a lot of information about the essays which might also be useful for organization scoring, but which the heuristic systems are unable to exploit. To remedy the problem, we introduce three learning-based systems which abstract the additional information we produced in three different ways. In each system, we use the SVM^{light} (Joachims, 1999) implementation of regression support vector machines (SVMs) (Cortes and Vapnik, 1995) to train a regressor because SVMs have been frequently and successfully applied to a variety of NLP problems.

6.1 Linear Kernel

Owing to the different ways we presented of combining the scores of an essay's nearest neighbors, the paragraph label sequence alignment approach has three variants, and its sentence label sequence alignment counterpart has nine. Unfortunately, these heuristic approaches suffer from two major weaknesses. First, it is not intuitively clear which of these 12 ways for predicting an essay's organization score is clearly better than the others. Second, it is not clear that the k nearest neighbors of an essay will always be similar to it with respect to organization score. While we do expect the alignment scores between good essays with reasonable paragraph sequences to be high, poorly organized essays by their nature have more random paragraph sequences. Hence, we have no intuition about the k nearest neighbors of a poor essay, as it may have as high an alignment score with another poorly organized essay as with a good essay.

Our solution to these problems is to use the organization scores obtained by the 12 heuristic variants as features in a linear kernel SVM learner. We believe that using the estimates given by all the 12 variants of the two heuristic approaches rather than only one of them addresses the first weakness mentioned above. The second weakness, on the other hand, is addressed by treating the organization score predictions obtained by the nearest neighbor methods as features for an SVM learner rather than as estimates of an essay's organization score.

The approach we have just described, however, does not exploit the full power of linear kernel

SVMs. One strength of linear kernels is that they make it easy to incorporate a wide variety of different types of features. In an attempt to further enhance the prediction capability of the SVM regressor, we will provide it with not only the 12 features derived from the heuristic-based approaches, but also with two additional types of features.

First, to give our learner more direct access to the information we used to heuristically predict essay scores, we can extract *paragraph label subsequences*[3] from each essay and use them as features. To illustrate the intuition behind these features, consider two paragraph subsequences: Introduction–Body and Rebuttal–Introduction. It is fairly typical to see the first subsequence, I–B, at the beginning of a good essay, so its occurrence should give us a small amount of evidence that the essay it occurs in is well-organized. The presence of the second subsequence, R–I, however, should indicate that its essay's organization is poor because, in general, a good essay should not give a Rebuttal before an Introduction. Because we can envision subsequences of various lengths being useful, we create a binary presence or absence feature in the linear kernel for each paragraph subsequence of length 1, 2, 3, 4, or 5 appearing in the training set.

Second, we employ *sentence label subsequences* as features in the linear kernel. Recall that when describing our alignment-based nearest neighbor organization score prediction methods, we noted that an essay's organization score may be partially attributable to how well the sentences within its paragraphs are organized. For example, if one of an essay's paragraphs contains the sentence label subsequence Main Idea–Elaboration–Support–Conclusion this gives us some evidence that the essay is overall well-organized since one of its component paragraphs contains this reasonably-organized subsequence. An essay with a paragraph containing the subsequence Conclusion–Support–Thesis–Rebuttal, however, is likely to be poorly organized because this is a poorly-organized subsequence. Since sentence label subsequences of differing lengths may be useful for score prediction, we create a binary presence or absence feature for each sentence label subsequence of length 1, 2, 3, 4, or 5 in the training set.

While the number of nearest neighbor features is manageable, the presence of a large number of features can sometimes confuse a learner. For that reason, we do feature selection on the two types of subsequence features, selecting only 100 features for each type that has the highest information gain (see Yang and Pedersen (1997) for details). We call the system resulting from the use of these three types of features Rl_{nps} because it uses **R**egression with **l**inear kernel to predict essay scores, and it uses **n**earest neighbor, **p**aragraph subsequence, and **s**entence subsequence features.

6.2 String Kernel

In a traditional learning setting, the feature set employed by an off-the-shelf learning algorithm typically consists of *flat* features (i.e., features whose values are discrete- or real-valued, as the ones described in the Linear Kernel subsection). Advanced machine learning algorithms such as SVMs, on the other hand, have enabled the use of *structured* features (i.e., features whose values are structures such as parse trees and sequences), owing to their ability to employ *kernels* to efficiently compute the similarity between two potentially complex structures.

Perhaps the most obvious advantage of employing structured features is *simplicity*. To understand this advantage, consider learning in a traditional setting. Recall that we can only employ flat features in this setting, as we did with the linear kernel. Hence, if we want to use information from a parse tree as features, we will need to design heuristics to extract the desired parse-based features from parse trees. For certain tasks, designing a good set of heuristics can be time-consuming and sometimes difficult. On the other hand, SVMs enable a parse tree to be employed directly as a structured feature, obviating the need to design heuristics to extract information from potentially complex structures. However, structured features have only been applied to a handful of NLP tasks such as semantic role labeling (Moschitti, 2004), syntactic parsing and named entity identification (Collins and Duffy, 2002), relation extraction (Bunescu and Mooney, 2005), and coreference resolution (Versley et al., 2008). Our goal here is to explore this rarely-exploited capability of SVMs for the task of essay scoring.

[3] Note that a subsequence is not necessarily contiguous.

While the vast majority of previous NLP work on using structured features have involved tree kernels, we employ a kernel that is rarely investigated in NLP: *string kernels* (Lodhi et al., 2002). Informally, a string kernel aims to efficiently compute the similarity between two strings (or sequences) of symbols based on the similarity of their subsequences. We apply string kernels to essay scoring as follows: we represent each essay using its paragraph function label sequence, and employ a string kernel to compute the similarity between two essays based on this representation. Typically, a string kernel takes as input two parameters: K (which specifies the length of the subsequences in the two strings to compare) and λ (which is a value between 0 and 1 that specifies whether matches between non-contiguous subsequences in the two strings should be considered as important as matches between contiguous subsequences). In our experiments, we select values for these parameters in a somewhat arbitrary manner. In particular, since λ ranges between 0 and 1, we simply set it to 0.5. For K, since in the flat features we considered all paragraph label sequences of lengths from 1 to 5, we again take the middle value, setting it to 3. We call the system using this kernel Rs because it uses a **R**egression SVM with a **s**tring kernel to predict essay scores.

6.3 Alignment Kernel

In general, the purpose of a kernel function is to measure the similarity between two examples. The string kernel we described in the previous subsection is just one way of measuring the similarity of two essays given their paragraph sequences. While this may be the most obvious way to use paragraph sequence information from a machine learning perspective, our earlier use of the Needleman-Wunsch algorithm suggests a more direct way of extracting structured information from paragraph sequences.

More specifically, recall that the Needleman-Wunsch algorithm finds an optimal alignment between two paragraph sequences, where an optimal alignment is defined as an alignment having the highest possible alignment score. The optimal alignment score can be viewed as another similarity measure between two essays. As such, with some slight modifications, the alignment score between two paragraph sequences can be used as the kernel value for an Alignment Kernel.[4] We call the system using this kernel Ra because it uses a **R**egression SVM with an **a**lignment kernel to predict essay scores.

6.4 Combining Kernels

Recall that the flat features are computed using a linear kernel, while the two types of structured features are computed using string and alignment kernels. If we want our learner to make use of more than one of these types of features, we need to employ a *composite* kernel to combine them. Specifically, we define and employ the following composite kernel:

$$K_c(F_1, F_2) = \frac{1}{n}\sum_{i=1}^{n} K_i(F_1, F_2),$$

where F_1 and F_2 are the full set of features (containing both flat and structured features) that represent the two essays under consideration, K_i is the ith kernel we are combining, and n is the number of kernels we are combining. To ensure that each kernel under consideration contributes equally to the composite kernel, each kernel value $K_i(F_1, F_2)$ is normalized so that its value falls between 0 and 1.

7 Evaluation

7.1 Evaluation Metrics

We designed three evaluation metrics to measure the error of our organization scoring system. The simplest metric, $S1$, is perhaps the most intuitive. It measures the frequency at which a system predicts the wrong score out of the seven possible scores. Hence, a system that predicts the right score only 25% of the time would receive an $S1$ score of 0.75.

The $S2$ metric is slightly less intuitive than $S1$, but no less reasonable. It measures the average distance between the system's score and the actual score. This metric reflects the idea that a system that estimates scores close to the annotator-assigned scores should be preferred over a system whose estimations are further off, even if both systems estimate the correct score at the same frequency.

Finally, the $S3$ evaluation metric measures the average square of the distance between a system's

[4] In particular, we note that for theoretical reasons, a kernel function must always return a non-negative value. The alignment score function does not have this property, so we increase all alignment scores until their theoretical minimum value is 0.

organization score estimations and the annotator-assigned scores. The intuition behind this system is that not only should we prefer a system whose estimations are close to the annotator scores, but we should also prefer one whose estimations are not too frequently very far away from the annotator scores. These three scores are given by:

$$\frac{1}{N}\sum_{A_i \neq E_i} 1, \quad \frac{1}{N}\sum_{i=1}^{N} |A_i - E_i|, \quad \frac{1}{N}\sum_{i=1}^{N} (A_i - E_i)^2,$$

where A_i and E_i are the annotator assigned and system estimated scores respectively for essay i, and N is the number of essays. Since many of the systems we have described assign test essays real-valued organization scores, to obtain E_i for system $S1$ we round the outputs of each system to the nearest of the seven scores the human annotators were permitted to assign (1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0).

To test our system, we performed 5-fold cross validation on our 1003 essay set, micro-averaging our results into three scores corresponding to the three scoring metrics described above.

7.2 Results and Discussion

The average baseline. As mentioned before, there is no standard baseline for organization modeling against which we can compare our systems. To start with, we employ a simple "average" baseline. Avg computes the average organization score of essays in the training set and assigns this score to each test set essay. Results of this baseline are shown in row 1 of Table 6. Though simple, this baseline is by no means easy-to-beat, since 41% of the essays have a score of 3, and 96% of the essays have a score that is within one point of 3.

Heuristic baselines. Recall that we have 12 versions of the two heuristic approaches to organization prediction. Space limitations preclude a discussion of the results of all these versions, so instead, to obtain the strongest baseline results, we show only the best results achieved by the three versions based on aligning paragraph label sequences in row 2 (H_p) and the best results achieved by the nine versions based on aligning sentence label sequences in row 3 (H_s) of Table 6. It is clear from the results that the H_p systems yielded the best baseline predictions under all three scoring metrics, performing significantly better than both the Avg and H_s systems ($p < 0.01$) with respect to the $S2$ and $S3$ metrics, but its $S1$ performance is less significant with respect to Avg ($p < 0.1$) and is indistinguishable at even the $p < 0.1$ level from H_s.[5] In general, however, it appears to be the case that systems based on aligning paragraph label sequences achieve better results than systems that attempt to align sentence label sequences.

	System	$S1$	$S2$	$S3$
1	Avg	.585	.412	.348
2	H_p	.548	.339	.198
3	H_s	.575	.397	.329
4	Rl_{nps}	.520	.331	.186
5	Rs	.577	.369	.222
6	Ra	.686	.519	.429
7	Rls_{nps}	.534	.332	.187
8	Rla_{nps}	.541	.332	.178
9	Rsa	.517	.325	.177
10	$Rlsa_{nps}$	.517	.323	.175

Table 6: System Performance

Learning-based approaches. Rows 4–6 of Table 6 show the results we obtained using each of the three single-kernel systems. When compared to the best baseline, these results suggest that H_p is a pretty good heuristic approach to organization scoring. In fact, only one of these three learning-based systems (Rl_{nps}) performs better than H_p under the three scoring metrics, and in each case, the difference between the two is not significant even at $p < 0.1$. This suggests that, even though Rl_{nps} performs slightly better than H_p, the only major benefit we have obtained by using the linear kernel is that it has made it unnecessary for us to choose between the 12 proposed heuristic systems.

Considering that the second best one-kernel system, Rs, does not have access to any of the nearest neighbor features, which have already proven useful, its performance seems reasonably good in that its performance is at least better than the Avg system. This suggests that, even though Rs does not perform exceptionally, it is extracting some useful information for organization scoring from the heuristically assigned paragraph label sequences. The best one-kernel system, Rl_{nps}, however, is sig-

[5] All significance tests are two-tailed paired t-tests.

nificantly better than Rs with respect to all three scoring metrics, with $p < 0.1$ for $S1$ and $p < 0.05$ for $S2$ and $S3$. By contrast, it initially appears that the alignment kernel is not extracting any useful information from these paragraph sequences at all, since its $S1$, $S2$, and $S3$ scores are all much worse than all of the baseline systems. The second best one-kernel system Rs performs significantly better than Ra at $p < 0.01$ for all three scoring metrics.

Next, we explore the impact of composite kernels, which allow our learners to make use of multiple types of flat and structured features. Specifically, the results shown in rows 7–9 are obtained by combining two kernels at a time. These experiments reveal the surprising result that the two worst performing single-kernel systems, Rs and Ra, when combined into Rsa, yield the best two-kernel system results, which are significant with respect to the best one-kernel system results under $S3$ at $p < 0.1$. This result suggests that these two different methods of extracting information from paragraph sequences provide us with different kinds of evidence useful for organization scoring, although neither method by itself was exceptionally useful. Though Rsa does not have any access to nearest neighbor information, it still performs significantly better than H_p at $p < 0.05$ under $S1$ and $S3$.

While we have already pointed out that Rsa is the best composite two-kernel system, it is not clear which of Rls_{nps} and Rla_{nps} is second-best. Neither system consistently performs better than the other under all three scoring metrics, and the differences between them are not significant even at $p < 0.1$. It is clear only that Rsa is better than both, as its scores are statistically significantly better at $p < 0.01$ with respect to Rls_{nps} and Rla_{nps} under at least one of the three scoring metrics in each case.

Finally, in the last row of Table 6, we combine all three kernels into one SVM learner. The most important lesson we learn from this experiment is that each of the three kernels provides the learner with a different kind of useful information, so that a composite kernel using all three sources of information performs better than any system using fewer kernels. Although the improvements over the best two-kernel system (Rsa) and one-kernel system (Rl_{nps}) are small, they are still statistically significant at $p < 0.1$ under one of the scoring metrics, $S3$. When we compare this combined system to the best baseline (H_p), we discover the improvements derived from the three-kernel system are significant improvements over it at $p < 0.05$ and $p < 0.01$ with respect to $S1$ and $S3$ respectively.

Feature analysis. To better understand which of the three flat features (nearest neighbors, paragraph label sequences, or sentence label sequences) contributes the most to the linear kernel portion of the systems' performances, we analyze the three feature types on Rl_{nps} using the backward elimination feature selection algorithm. First, we remove each of the three feature groups independently from the Rl_{nps}'s feature set and determine which of the three removals yields the best performance according to each scoring metric. Next, among the remaining two feature groups, we repeat the same step, removing each of the two groups independently from the feature set to determine which of the two removals yields the best performance.

While space limitations preclude showing the actual numbers, the trend is consistent among all three scoring metrics: the first feature type to remove is paragraph sequences (meaning that they are the least important) and the last to remove is the nearest neighbor features. Nevertheless, performance always drops when a feature type is removed, indicating that all three feature types contribute positively to overall performance. The fact that flat paragraph sequence features proved to be least useful highlights the importance of the structured methods we presented for using paragraph sequence information.

8 Conclusions

We have investigated the relatively less-studied problem of modeling the organization in student essays. The contributions of our work include the novel application of two techniques from bioinformatics and machine learning — sequence alignment and string kernels, as well as the introduction of alignment kernels — to essay scoring. We showed that each technique makes a significant contribution to a scoring system, and we hope that this work will increase awareness of these powerful techniques among NLP researchers. Finally, to stimulate work on this problem, we make our corpus of annotated essays available to other researchers.

Acknowledgments

We thank the three reviewers for their comments. Our six annotators, Andrew Hubbs, Karin Khoo, Jayne Koath, Christopher Maier, Andrew Mallon, and Cory Thornton, all deserve numerous thanks, because without the countless hours they each spent annotating hundreds of essays, none of the research described in this paper would have been possible.

References

Yigal Attali and Jill Burstein. 2006. Automated essay scoring with e-rater V.2. *Journal of Technology, Learning, and Assessment*, 4(3).

Regina Barzilay and Mirella Lapata. 2005. Modeling local coherence: An entity-based approach. In *Proceedings of the ACL*, pages 141–148.

Regina Barzilay and Lillian Lee. 2002. Bootstrapping lexical choice via multiple-sequence alignment. In *Proceedings of EMNLP*, pages 164–171.

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

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *HLT-NAACL 2004: Main Proceedings*, pages 113–120.

Razvan Bunescu and Raymond Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of HLT/EMNLP*, pages 724–731.

Jill Burstein, Martin Chodorow, and Claudia Leacock. 2004. Automated essay evaluation: The *Criterion* online writing evaluation service. *AI Magazine*, 25(3), 27–36.

Jill Burstein, Daniel Marcu, and Kevin Knight. 2003. Finding the write stuff: Automatic identification of discourse structure in student essays. *IEEE Intelligent Systems*, 18(1):32–39.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the ACL*, pages 263–270.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.

Scott Elliot. 2001. IntelliMetric: From here to validity. Paper presented at the annual meeting of the American Educational Research Association, Seattle, WA.

Micha Elsner, Joseph Austerweil, and Eugene Charniak. 2007. A unified local and global model for discourse coherence. In *NAACL HLT 2007: Proceedings of the Main Conference*, pages 436–443.

Sylviane Granger, Estelle Dagneaux, Fanny Meunier, and Magali Paquot. 2009. *International Corpus of Learner English (Version 2)*. Presses universitaires de Louvain.

Derrick Higgins, Jill Burstein, Daniel Marcu, and Claudia Gentile. 2004. Evaluating multiple aspects of coherence in student essays. In *HLT-NAACL 2004: Main Proceedings*, pages 185–192.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In Bernhard Scholkopf and Alexander Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 44–56. MIT Press.

Thomas K. Landauer, Darrell Laham, and Peter W. Foltz. 2003. Automated scoring and annotation of essays with the Intelligent Essay Assessor^{TM}. In *Automated Essay Scoring: A Cross-Disciplinary Perspective*, pages 87–112.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Christopher J. C. H. Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444.

Eleni Miltsakaki and Karen Kukich. 2004. Evaluation of text coherence for electronic essay scoring systems. *Natural Language Engineering*, 10(1):25–55.

Alessandro Moschitti. 2004. A study on convolution kernels for shallow statistic parsing. In *Proceedings of the ACL*, pages 335–342.

Saul Ben Needleman and Christian Dennis Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, March.

Mark Shermis and Jill Burstein. 2003. *Automated Essay Scoring: A Cross-Disciplinary Perspective*. Lawrence Erlbaum Associates, Inc., Mahwah, NJ.

Mark Shermis, Jill Burstein, Derrick Higgins, and Klaus Zechner. 2010. Automated essay scoring: Writing assessment and instruction. In *International Encyclopedia of Education (3rd edition)*, pages 20–26.

Tony Silva. 1993. Toward an understanding of the distinct nature of L2 writing: The ESL research and its implications. 27(4):657–677.

Radu Soricut and Daniel Marcu. 2006. Discourse generation using utility-trained coherence models. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 803–810.

Yannick Versley, Alessandro Moschitti, Massimo Poesio, and Xiaofeng Yang. 2008. Coreference systems based on kernels methods. In *Proceedings of COLING*, pages 961–968.

Yiming Yang and Jan O. Pedersen. 1997. A comparative study on feature selection in text categorization. In *Proceedings of ICML*, pages 412–420.

Evaluating Models of Latent Document Semantics in the Presence of OCR Errors

Daniel D. Walker, William B. Lund, and Eric K. Ringger
Brigham Young University
Provo, Utah, USA
danl4@cs.byu.edu, bill_lund@byu.edu, ringger@cs.byu.edu

Abstract

Models of latent document semantics such as the mixture of multinomials model and Latent Dirichlet Allocation have received substantial attention for their ability to discover topical semantics in large collections of text. In an effort to apply such models to noisy optical character recognition (OCR) text output, we endeavor to understand the effect that character-level noise can have on unsupervised topic modeling. We show the effects both with document-level topic analysis (document clustering) and with word-level topic analysis (LDA) on both synthetic and real-world OCR data. As expected, experimental results show that performance declines as word error rates increase. Common techniques for alleviating these problems, such as filtering low-frequency words, are successful in enhancing model quality, but exhibit failure trends similar to models trained on unprocessed OCR output in the case of LDA. To our knowledge, this study is the first of its kind.

1 Introduction

As text data becomes available in massive quantities, it becomes increasingly difficult for human curators to manually catalog and index modern document collections. To aid in the automation of such tasks, algorithms can be used to create models of the latent semantics present in a given corpus. One example of this type of analysis is document clustering, in which documents are grouped into clusters by topic. Another type of topic analysis attempts to discover finer-grained topics—labeling individual words in a document as belonging to a particular topic. This type of analysis has grown in popularity recently as inference on models containing large numbers of latent variables has become feasible.

The modern explosion of data includes vast amounts of historical documents, made available by means of Optical Character Recognition (OCR), which can introduce significant numbers of errors. Undertakings to produce such data include the Google Books, Internet Archive, and HathiTrust projects. In addition, researchers are having increasing levels of success in digitizing hand-written manuscripts (Bunke, 2003), though error rates remain much higher than for OCR. Due to their nature, these collections often lack helpful meta-data or labels. In the absence of such labels, unsupervised machine learning methods can reveal patterns in the data.

Finding good estimates for the parameters of models such as the mixture of multinomials document model (Walker and Ringger, 2008) and the Latent Dirichlet Allocation (LDA) model (Blei et al., 2003) requires accurate counts of the occurrences and co-occurrences of words. Depending on the age of a document and the way in which it was created, the OCR process results in text containing many types of noise, including character-level errors, which distort these counts. It is obvious, therefore, that model quality must suffer, especially since unsupervised methods are typically much more sensitive to noise than supervised methods. Good supervised learning algorithms are substantially more immune to spurious patterns in the data created by noise for the following reason: under the mostly reasonable assumption that the process contributing the noise operates independently from the class labels, the noise in the features will not correlate well with the class labels, and the algorithm will learn

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 240–250,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

to ignore those features arising from noise. Unsupervised models, in contrast, have no grounding in labels to prevent them from confusing patterns that emerge by chance in the noise with the "true" patterns of potential interest. For example, even on clean data, LDA will often do poorly if the very simple feature selection step of removing stop-words is not performed first. Though we expect model quality to decrease, it is not well understood how sensitive these models are to OCR errors, or how quality deteriorates as the level of OCR noise increases.

In this work we show how the performance of unsupervised topic modeling algorithms degrades as character-level noise is introduced. We demonstrate the effect using both artificially corrupted data and an existing real-world OCR corpus. The results are promising, especially in the case of relatively low word error rates (e.g. less than 20%). Though model quality declines as errors increase, simple feature selection techniques enable the learning of relatively high quality models even as word error rates approach 50%. This result is particularly interesting in that even humans find it difficult to make sense of documents with error rates of that magnitude (Munteanu et al., 2006).

Because of the difficulties in evaluating topic models, even on clean data, these results should not be interpreted as definitive answers, but they do offer insight into prominent trends. For example, properties of the OCR data suggest measures that can be taken to improve performance in future work. It is our hope that this work will lead to an increase in the usefulness of collections of OCRed texts, as document clustering and topic modeling expose useful patterns to historians and other interested parties.

The remainder of the paper is outlined as follows. After an overview of related work in Section 2, Section 3 introduces the data used in our experiments, including an explanation of how the synthetic data were created and of some of their properties. Section 4 describes how the experiments were designed and carried out, and gives an analysis of the results both empirically and qualitatively. Finally, conclusions and future work are presented in Section 5.

2 Related Work

Topic models have been used previously to process documents digitized by OCR, including eighteenth-century American newspapers (Newmann and Block, 2006), OCRed editions of *Science* (Blei and Lafferty, 2006), OCRed NIPS papers (Wang and McCallum, 2006), and books digitized by the Open Content Alliance (Mimno and Mccallum, 2007). Most of this previous work ignores the presence of OCR errors or attempts to remove corrupted tokens with special pre-processing such as stop-word removal and frequency cutoffs. Also, there are at least two instances of using topic modeling to improve the results of an OCR algorithm (Wick et al., 2007; Farooq et al., 2009).

Similar evaluations to ours have been conducted to assess the effect of OCR errors on supervised document classification (Taghva et al., 2001; Agarwal et al., 2007), information retrieval (Taghva et al., 1994; Beitzel et al., 2003), and a more general set of natural language processing tasks (Lopresti, 2008). Results suggest that in these supervised tasks OCR errors have a minimal impact on the performance of the methods employed, though it has remained unclear how well these results transfer to unsupervised methods.

3 Data

We conducted experiments on synthetic and real OCR data. As a real-world dataset, we used a corpus consisting of 604 of the Eisenhower World War II communiqués (Jordan, 1945; Lund and Ringger, 2009). These documents relate the daily progress of the Allied campaign from D-Day until the German surrender. They were originally produced as telegrams and were distributed as mimeographed copies. The quality of the originals is often quite poor, making them a challenging case for OCR engines. The communiqués have been OCRed using three popular OCR engines: ABBYY FineReader (ABBYY, 2010), OmniPage Pro (Nuance Communications, Inc., 2010), and Tesseract (Google, Inc., 2010). In addition, the curator of the collection has created a "gold standard" transcription, from which it is possible to obtain accurate measures of average document word error rates (WER) for each engine, which are: 19.9%, 30.4%, and 50.1% respectively.

While the real-world data is attractive as an example of just the sort of data that the questions addressed here apply to, it is limited in size and scope. All of the documents in the Eisenhower corpus discuss the fairly narrow topic of troop movements and battle developments taking place at the end of World War II. Neither the subject matter nor the means of conveyance allowed for a large or diverse vocabulary of discourse.

In an attempt to generalize our results to larger and more diverse data, we also ran experiments using synthetic OCR data. This synthetic data was created by corrupting "clean" datasets, adding character-level noise. The synthetic data was created by building a noise model based on mistakes made by the worst performing OCR engine on the Eisenhower dataset, Tesseract.

To construct the noise model, a character-level alignment between the human transcribed Eisenhower documents and the OCR output was first computed. From this alignment, the contingency table $\mathbf{M}^d$ was generated such that $\mathbf{M}^d_{x,y}$ was the count of the instances in which a character x in the transcript was aligned with a y in the OCR output. The rows in this matrix were then normalized so that each represented the parameters of a categorical distribution, conditioned on x. To parameterize the amount of noise being generated, the $\mathbf{M}^d$ matrix was interpolated with an identity matrix $\mathbf{I}$ using a parameter γ so that the final interpolated parameters $\mathbf{M}^i$ were calculated with the formula $\mathbf{M}^i = \gamma\mathbf{M}^d + (1-\gamma)\mathbf{I}$. So that at $\gamma = 0$, $\mathbf{M}^i = \mathbf{I}$ and no errors were introduced. At $\gamma = 1.0$, $\mathbf{M}^i = \mathbf{M}^d$, and we would expect to see characters corrupted at the same rate as in the output of the OCR engine.

We then iterated over each document, choosing a new (possibly the same) character y_l for each original character x_l according to the probability distribution $p(y_l = w'|x_l = w) = M^i_{w,w'}$. Our process was a one-substitution algorithm, as we did not include instances of insertions or deletions, consequently words were changed but not split or deleted. This allowed for a more straightforward calculation of word error rate. Segmentation errors can still occur in the learning stage, however, as the noise model sometimes replaced alphabet characters with punctuation characters, which were treated as delimiters by our tokenizer.

Dataset	**|D|**	K	**# Types**	**# Tokens**
20 News	19997	20	107211	2261805
Reuters	11367	81	29034	747458
Enron	4935	32	60495	2063667
Eisenhower	604	N/A	8039	76674

Table 1: Summary of test dataset characteristics. $|D|$ is the number of documents in the dataset. K is the number of human-labeled classes provided with the dataset.

We chose three datasets to corrupt: 20 Newsgroups (Lang, 1995), Reuters 21578 (Lewis, 1997), and the LDC-annotated portion of the Enron e-mail archive (Berry et al., 2007). Each of these datasets were corrupted at values $\gamma = i * 0.01$ for $i \in (0, 13)$. At this point, the word error rate of the corrupted data was near 50% and, since this was approximately the WER observed for the worst OCR engine on the real-world data, we chose to stop there. The word error rate was calculated during the corruption process. Here is an example sentence corrupted at two γ values:

$\gamma = 0.000$ I am also attaching the RFP itself.
$\gamma = 0.02$ I am also attachEng the RFP itself.
$\gamma = 0.10$ I Jm alAo attaching the RFP itself.

Table 3 shows some basic statistics for the datasets. The values shown are for the "clean" versions of the data. For an example of how noise and pre-processing techniques affect these counts see Section 4.1.

It is interesting to note that the word error rates produced by the noise model appear to be significantly higher than first expected. One might assume that the WER should increase fairly steadily from 0% at $\gamma = 0$ to about 50% (the error rate achieved by the Tesseract OCR engine on the Eisenhower dataset) at $\gamma = 1$. There are at least two sources for the discrepancy. First, the vocabulary of the Eisenhower dataset does not match well with that of any of the source datasets from which the synthetic data were generated. This means that the word and character distributions are different and so the error rates will be as well. Secondly, whereas our technique gives the same probability of corruption to all instances of a given character, errors in true OCR output are bursty and more likely to be concentrated in specific tokens, or regions, of a document. This is because most sources of noise do not affect document images uniformly. Also, modern OCR engines do not operate at just the character level. They incorporate dictionaries and language models to prevent them from positing words that are highly unlikely. As a consequence, an OCR engine is much more likely to either get a whole word correct, or to miss it completely, concentrating multiple errors in a single word. This is the difference between 10 errors in a single word, which only contributes 1 to the numerator of the WER formula and 10 errors spread across 10 different words, which contributes 10 to the numerator. Furthermore, because content bearing words tend to be relatively rare, language models are poorer for them than for frequent function words, meaning that the words most correlated with semantics are also the most likely to be corrupted by an OCR engine.

An example of this phenomenon is easy to find. In the Enron corpus, there are 165,871 instances of the word "the" and 102 instances of the string "thc". Since "c" has a high rate of confusion with "e", we would expect at least some instances of "the" to be corrupted to "thc" by the error model. At $\gamma = 0.03$, there are 156,663 instances of the word "the" and 513 instances of "thc". So, the noise model converts "the" to "thc" roughly 0.3% of the time. In contrast, there are no instances of "thc" in the Tesseract OCR output even though there are 5186 instances of "the" in the transcription text, and so we would expect approximately 16 occurrences of "thc" if the errors introduced by the noise model were truly representative of the errors in the actual OCR output.

Another interesting property of the noise introduced by actual OCR engines and our synthetic noise model is the way in which this noise affects words distributions. This is very important, since word occurrence and co-occurrence counts are the basis for model inference in both clustering and topic modeling. As mentioned previously, one common way of lessening the impact of OCR noise when training topic models over OCRed data is to apply a frequency cutoff filter to cull words that occur fewer than a certain number of times. Figures 1 and 2 show the number of word types that are culled from the synthetic 20 Newsgroups OCR data and the Eisenhower OCR data, respectively, at various levels of noise. Note that the cutoff filters use a strict "less than", so a frequency cutoff of 2 eliminates only words that occur once in the entire dataset. Also, these series are additive, as the words culled with a frequency cutoff of 2 are a subset of those culled with a frequency cutoff of $j > 2$.

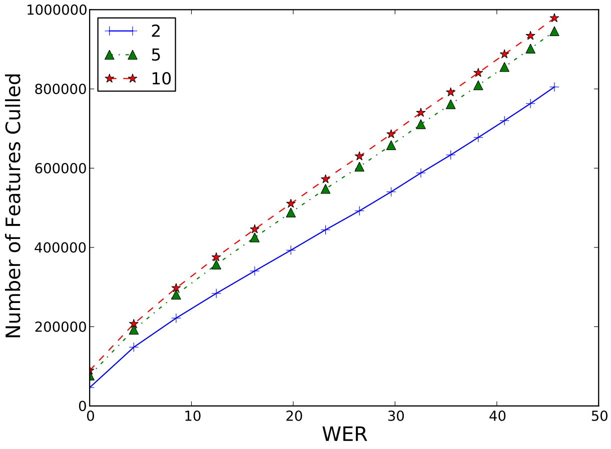


Figure 1: The number of word types culled with frequency cutoff filters applied to the 20 Newsgroups data with various levels of errors introduced.

In both cases, it is apparent that by far the largest impact that noise has is in the creation of singletons. It seems that the most common corruptions in these scenarios is the creation of one-off word types through a unique corruption of a (most likely rare) word. This means that it is unlikely that enough evidence will be available to associate, through similar contexts, the original word and its corrupted forms.

Due to the fact that most clustering and topic models ignore the forms of word tokens (the characters that make them up), and only take into account word identities, we believe that the similarity in the way in which real OCR engines and our synthetic OCR noise model distort word distributions is sufficient evidence to support the use of the synthetic data until larger and better real-world OCR datasets can be made available. Though the actual errors will take a different form, the character-level details of the errors are less relevant than the word distribution alterations for the models in question.

4 Experimental Results

We ran experiments on both the real and synthetic OCR data. In this section we explain our experi-

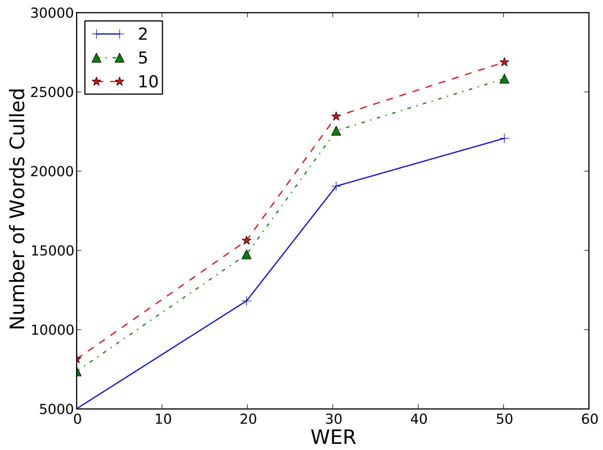


Figure 2: The number of word types culled with frequency cutoff filters applied to the transcript and three OCR engine outputs for the Eisenhower data.

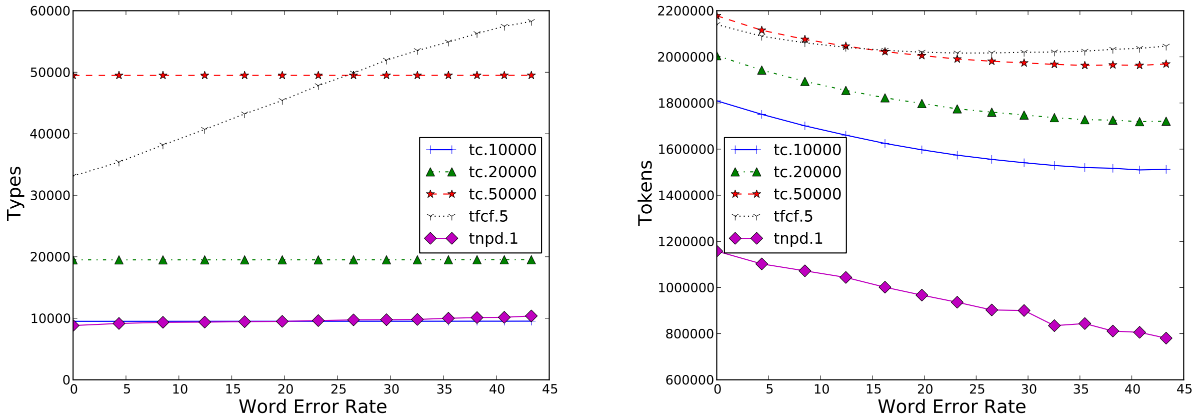
mental methodology and present both empirical and qualitative analyses of the results.

4.1 Methodology

For the synthetic OCR datasets, we ran clustering experiments using EM on a mixture of multinomials (c.f. (Walker and Ringger, 2008)). We specified the number of clusters to be the same as the number of classes provided with the data. Clusters were evaluated using several external cluster quality metrics which compare "gold standard" labels to those created through clustering. The metrics used were Variation of Information (VI) (Meilă, 2007), and the Adjusted Rand Index (ARI) (Hubert and Arabie, 1985). Other metrics were also calculated (e.g. the V-Measure (Rosenberg and Hirschberg, 2007), and Average Entropy (Liu et al., 2003)), but these results were excluded due to space constraints and the fact that their plots are similar to those shown. We did not cluster the Eisenhower data because of the absence of the class labels necessary for evaluation.

For both the synthetic and non-synthetic data we also trained LDA topic models (Blei et al., 2003) using Gibbs sampling. We used the implementation found in the MALLET software package (McCallum, 2002) with the option enabled to learn the priors during sampling as discussed by Wallach et al. (2009a). Each LDA model was trained on 90% of the documents in each dataset. The trained model was used to calculate an estimate of the marginal log-likelihood of the remaining 10% of the documents using the left-to-right algorithm (Wallach et al., 2009b). The number of topics used for each dataset was adjusted *a priori* according to the number of documents it contained. We used 100 topics for Enron and Eisenhower, 150 for Reuters, and 200 for 20 Newsgroups.

In addition to running experiments on the "raw" synthetic data, we also applied simple unsupervised feature selectors before training in order to evaluate the effectiveness of such measures in mitigating problems caused by OCR errors. For the topic modeling (LDA) experiments three feature selectors were used. The first method employed was a simple term frequency cutoff filter (TFCF), with a cutoff of 5 as in (Wang and McCallum, 2006). The next method employed was Term Contribution (TC), a feature selection algorithm developed for document clustering (Liu et al., 2003). Term contribution is parameterized by the number of word types that are to remain after selection. We attempted three values for this parameter, 10,000, 20,000, and 50,000. The final method we employed was a method called Top-N per Document (TNPD) (Walker and Ringger, 2010), which is a simple feature selection algorithm that first assigns each type in every document a document-specific score (e.g. its TF-IDF weight), and then selects words to include in the final vocabulary by choosing the N words with the highest score from each document in the corpus. We found that $N = 1$ gave the best results at the greatest reduction in word types. After the vocabulary is built, all words not in the vocabulary are culled from the documents. This does not mean that all documents contain only one word after feature selection, as the top word in one document may occur in many other documents, even if it is not the top word in those documents. Likewise, if two documents would both contribute the same word, then the second document makes no contribution to the vocabulary. This process can result in vocabularies with thousands of words, leaving sufficient words in each document for analysis. For the clustering experiments, initial tests showed little difference in the performance of the feature selectors, so only the TNPD selector was used. Figures 3(a) and 3(b) show how the various pre-processing methods affect word type and token



(a) The number of word types remaining after pre-processing.

(b) The number of word tokens remaining after pre-processing.

Figure 3: The effect of pre-processing on token and type counts for the 20 Newsgroups dataset at various error rates.

counts, respectively, for the 20 Newsgroups dataset. In contrast, without pre-processing the number of types scales from 107,211 to 892,983 and the number of tokens from 2,261,805 to 3,073,208.

Because all of these procedures alter the number of words and tokens in the final data, log-likelihood measured on a held-out set cannot be used to accurately compare the quality of topic models trained on pre-processed data, as the held-out data will contain many unknown words. If the held-out data is also pre-processed to only include known words, then the likelihood will be greater for those procedures that remove the most tokens, as the product that dominates the calculation will have fewer terms. If unknown words are allowed to remain, even a smoothed model will assign them very little probability and so models will be heavily penalized.

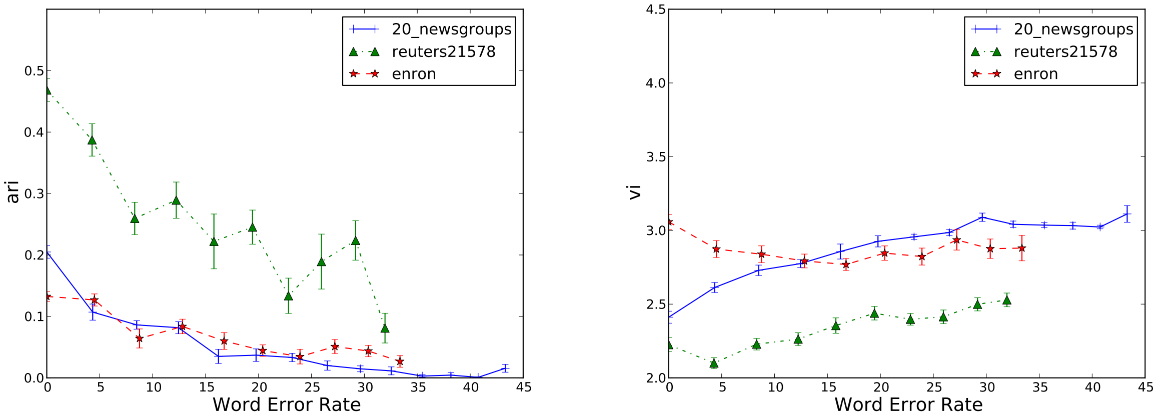
We use an alternative method for evaluating the topic models, discussed in (Griffiths et al., 2005), to avoid the aforementioned problems with an evaluation based on log-likelihood. Since the synthetic data is derived from datasets that have topical document labels, we are able to use the output from LDA in a classification problem with the word vectors for each document being replaced by the assigned topic vectors. This is equivalent to using LDA as a dimensionality reduction pre-process for document classification. A naive Bayes learner is trained on a portion of the topic vectors, labeled with the original document label, and then the classification accuracy on a held-out portion of the data is computed. Ten-fold cross-validation is used to control for sampling issues. The rationale behind this evaluation is that, even though the topics discovered by LDA will not correspond directly to the labels, there should at least be a high degree of correlation. Models that discover topical semantics that correlate well with the labels applied by humans will yield higher classification accuracies and be considered better models according to this metric.

To compensate for the randomness inherent in the algorithms, each experiment was replicated ten times. The results of these runs were averaged to produce the values reported here.

4.2 Empirical Analysis

Both the mixture of multinomials document model and LDA appear to be fairly resilient to character-level noise. Figures 4 and 5 show the results of the document clustering experiments with and without feature selection, respectively. Memory issues prevented the collection of results for the highest error rates on the Enron and Reuters data without feature selection.

With no pre-processing, the results are somewhat mixed. The Enron dataset experiences almost no quality degradation as the WER increases, yielding remarkably constant results according to the metrics. However, this may be an artifact of the relatively poor starting performance for this dataset, a result of the fact that the gold standard labels do not align



(a) Adjusted Rand Index results

(b) Variation of Information results (lower is better)

Figure 4: Results for the clustering experiments on the three synthetic datasets *without* feature selection.

well with automatically discovered patterns because they correspond to external events. In contrast, the Reuters data appears to experience drastic degradation in performance. Once feature selection occurs, however, performance remains much more stable as error rates increase.

Figure 6(a) shows the results of running LDA on the transcript and digitized versions of the Eisenhower dataset. Log-likelihoods of the held-out set are plotted with respect to the WER observed for each OCR engine. The results support the finding that the WER of the OCR engine that produced the data has a significant negative correlation with model quality. Unfortunately, it was not possible to compare the performance of the pre-processing methods on this dataset, due to a lack of document topic labels and the deficiencies of log-likelihood mentioned previously.

Figure 6(b) shows the results of the LDA topic-modeling experiments on the three "raw" synthetic datasets. Similar trends are observed in this graph. LDA experiences a marked degree of performance degradation, with all of the trend lines indicating a linear decrease in log-likelihood.

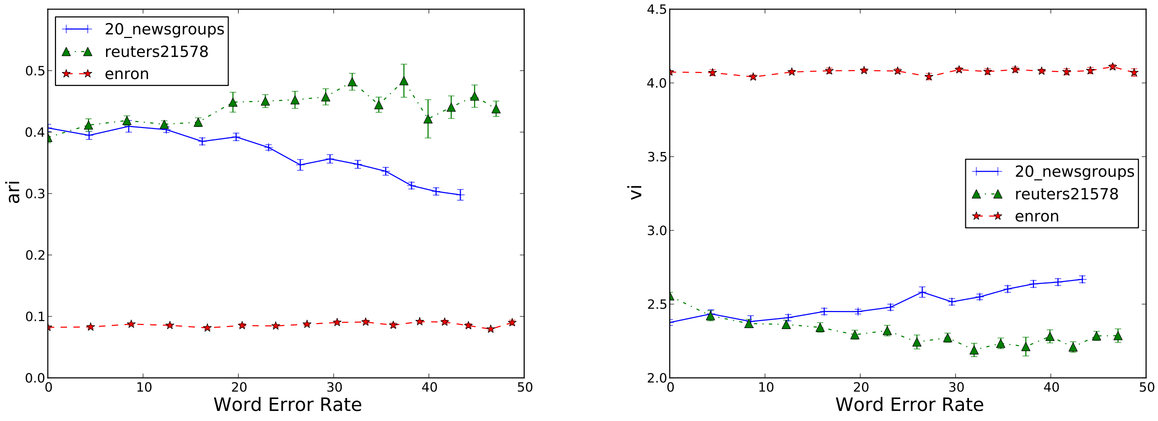
Figures 7(a) through 7(c) show the results of evaluating the various proposed pre-processing procedures in the context of topic modeling. In the graph "noop.0" represents no pre-processing, "tc.N" are the Term Contribution method parameterized to select N word types, "tfcf.5" is the term frequency cutoff filter with a cutoff of 5 and "tnpd.1" is the Top N per Document method with $N = 1$. The y-axis is the average of the results for 10 distinct trials, where the output for each trial is the average of the ten accuracies achieved using ten-fold cross-validation as described in Section 4.1.

Here, the cross-validation accuracy metric reveals a slightly different story. These results show that topic quality on both the raw and pre-processed noisy data degrades at a rate relative to the amount of errors in the data. That is, the difference in performance between two relatively low word error rates (e.g. 5% and 7% on the Reuters data) is small, whereas the differences between two high error rates (e.g. 30% and 32% on the Reuters data) can be relatively quite large.

While pre-processing does improve model quality, in the case of LDA this improvement amounts to a nearly constant boost; at high error rates quality is improved the same amount as at low error rates. Degradations in model quality, therefore, follow the same trends, occurring at mostly the same rate in pre-processed data as in the raw noisy data. This is in contrast to the clustering experiments where pre-processing virtually eliminates failure trends.

4.3 Qualitative Analysis

Higher values measured with automated metrics such as log-likelihood on a held-out set and the cross-validation classification metric discussed here



(a) Adjusted Rand Index results

(b) Variation of Information results (lower is better)

Figure 5: Results for the clustering experiments on the three synthetic OCR datasets *with* TNPD feature selection.

do not necessarily indicate superior topics according to human judgement (Chang et al., 2009). In order to provide a more thorough discussion of the relative quality of the topic models induced on the OCR data versus those induced on clean data, we sampled the results of several of the runs of the LDA algorithm. In Tables 2 and 3 we show the top words for the five topics with the highest learned topic prior (α in the LDA literature) learned during Gibbs sampling. This information is shown for the Reuters data first with no corruption and then at the highest error rate for which we have results for that data of 45% WER.

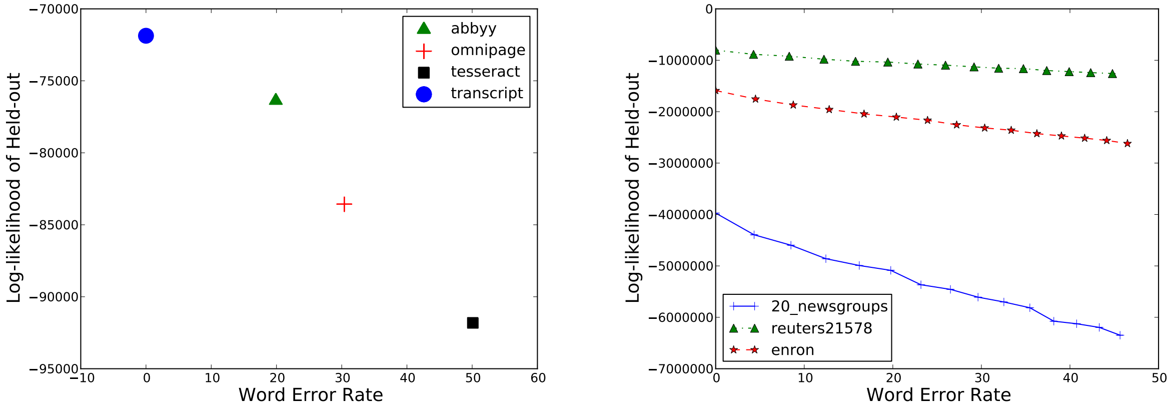
In general, there appears to be a surprisingly good correlation between the topics learned on the clean data and those learned on the corrupted data, given the high level of noise involved. The topics are generally cohesive, containing mostly terms relating to financial market news. However, the topics trained on the clean data, though all related to financial markets, are fairly distinctive. Topic 3, for example seems to be about fluctuations in stock prices, and Topics 106 and 34 about business acquisitions and mergers. The topics trained on the noisy data are fairly homogeneous and the differences between them are more difficult to identify.

In addition, it appears as though the first topic (topic 93) is not very coherent at all. This topic is significantly larger, in terms of the number of tokens assigned to it than the other topics shown in either table. In addition, the most probable words listed for the topic seem less cohesive than for the other topics. It contains many two-letter words that are likely a mixture of valid terms (e.g., stock exchange and ticker symbols, and parts of place names like “Rio de Janeiro”) and corruptions of real words. For example, even though there are no instances of “ts” as a distinct token in the clean Reuters data, it is in the list of the top 19 words for topic 93. This is perhaps due to the fact that “is” can easily be converted to “ts” by mistaking t for i.

It is also the case that, for most topics learned on the corrupted data, the most probable words for those topics tend to be shorter, on average, than for topics learned on clean data. We believe this is due to the fact that the processes used to add noise to the data (both real OCR engines and our synthetic noise model) are more likely to corrupt long words, especially in the case of the synthetic data which was created using a character-level noise model.

Examination of the data tends to corroborate this hypothesis, though even long words usually contain only a few errors. For example, in the 20 Newsgroups data there are 379 instances of the word “yesterday”, a long word that is not close to other English words in edit distance. When the data has been corrupted to a WER of 47.9%, however, there are only 109 instances of “yesterday” and 132 tokens that are within an edit distance of 1 from “yesterday”.

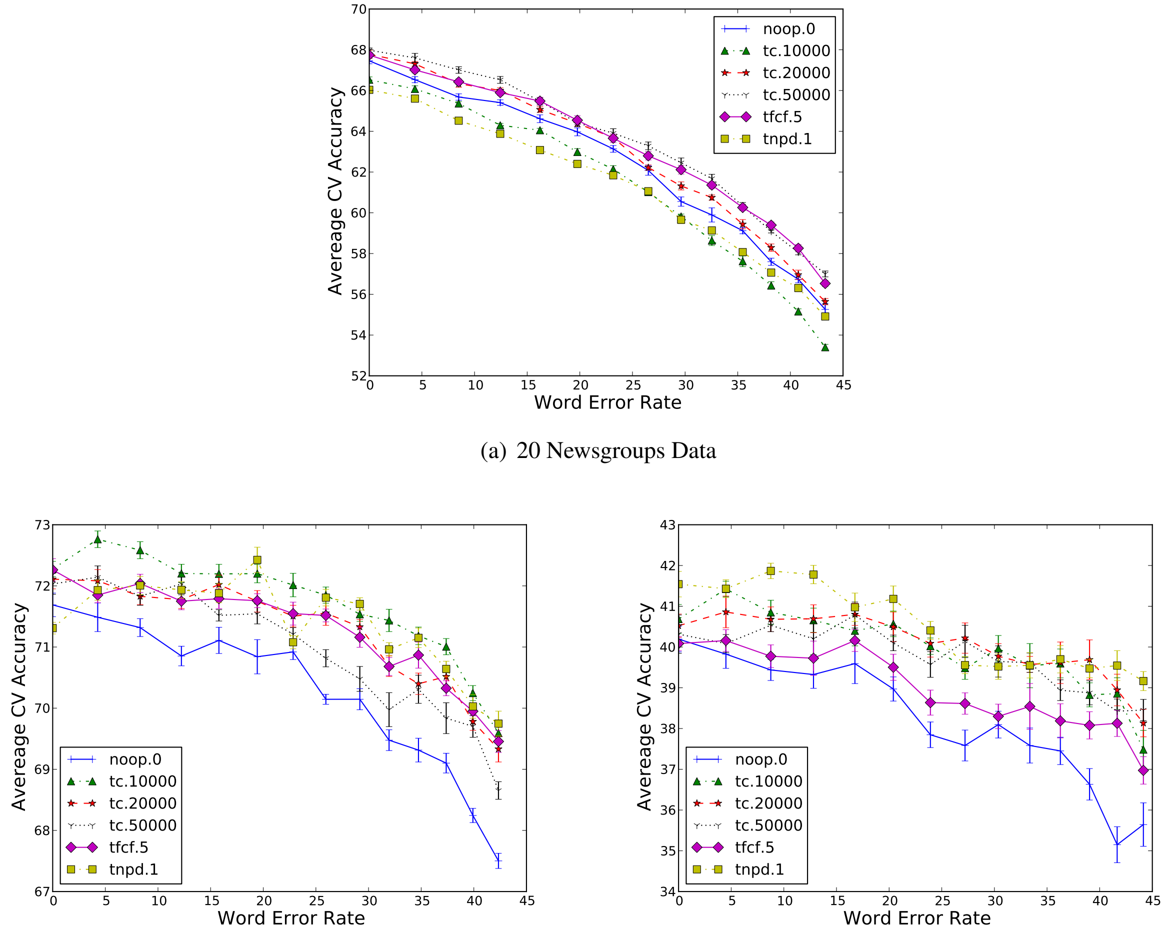
To some extent, we would expect to observe similar trends in real-world data. However, most OCR



(a) Eisenhower Communiqués

(b) Synthetic Data

Figure 6: Log-likelihood of heldout data for the LDA experiments.



(a) 20 Newsgroups Data

(b) Reuters Data

(c) Enron Data

Figure 7: Average ten-fold cross-validation accuracy for the LDA pre-processing experiments on the synthetic OCR data.

Topic	Words	Tokens
64	told, market, reuters, reuter, added, time, year, major, years, president, make, made, march, world, today, officials, industry, government, move	67159
3	year, pct, prices, expected, rise, lower, higher, demand, increase, due, fall, decline, current, high, end, added, level, drop, market	32907
106	reuter, corp, company, unit, sale, march, dlrs, mln, sell, subsidiary, acquisition, terms, group, april, purchase, acquired, products, division, business	22167
34	shares, dlrs, company, mln, stock, pct, share, common, reuter, corp, agreement, march, shareholders, buy, cash, outstanding, merger, acquire, acquisition	22668
7	mln, cts, net, shr, qtr, revs, reuter, avg, shrs, march, mths, dlrs, sales, st, corp, oct, note, year, april	18511

Table 2: Top words for the five topics with the highest α prior values found using MALLET for one run of LDA on the uncorrupted Reuters data.

Topic	Words	Tokens
93	reuter, march, pct, year, april, ed, market, er, told, es, st, end, ts, al, de, ng, id, sa, added	258932
99	company, pct, corp, shares, stock, dlrs, share, offer, group, reuter, mln, march, unit, stake, buy, cash, bid, sale, board	50377
96	mln, cts, net, shr, qtr, dlrs, revs, reuter, note, oper, avg, march, shrs, year, mths, st, sales, corp, oct	54659
141	mln, dlrs, year, net, quarter, share, company, billion, tax, sales, earnings, dlr, profit, march, income, ln, results, sale, corp	40475
53	pct, year, rose, rise, january, february, fell, march, index, december, month, figures, compared, reuter, rate, earlier, show, ago, base	22556

Table 3: Top words for the five topics with the highest α prior values found using MALLET for one run of LDA on the Reuters data corrupted with the data-derived noise model to a WER of 45%.

engines employ language models and dictionaries to attempt to mitigate OCR errors. As a result, given that a word recognition error has occurred in true OCR output, it is more likely to be an error that lies at an edit distance greater than one from the true word, or else it would have been corrected internally. For example, there are 349 instances of the word “yesterday” in the Eisenhower transcripts, and 284 instances in the Tesserect OCR output and only 5 tokens within an edit distance of one, meaning that 60 corruptions of this word contained more than one error, making up 90% of the errors for that word. However, many of these errors still contain most of the letters from the original word (e.g. “yesterdj.”, and “yestjkday”). In all cases, the corrupted versions of a given word are very rare, occurring usually only once or twice in the noisy output, making them useless features for informing a model.

5 Conclusions and Future Work

The primary outcome of these experiments is an understanding regarding when clustering and LDA topic models can be expected to function well on noisy OCR data. Our results imply that clustering methods should perform almost as well on OCR data as they do on clean data, provided that a reasonable feature selection algorithm is employed. The LDA topic model degraded less gracefully in performance with the addition of character level errors to its input, with higher error rates impacting model quality in a way that was apparent empirically in the log-likelihood and ten-fold cross-validation metrics as well as through human inspection of the produced topics. Pre-processing the data also helps model quality for LDA, yet still yields failure trends similar to those observed on unprocessed data.

We found it to be the case that even in data with high word error rates, corrupted words often share many characters in common with their uncorrupted form. This suggests an approach in which word similarities are used to cluster the unique corrupted versions of a word in order to increase the evidence available to the topic model during training time and improve model quality. As the quality of models increases on these noisy datasets, we anticipate a consequent rise in their usefulness to researchers and historians as browsing the data and mining it for useful patterns becomes more efficient and profitable.

Acknowledgments

We would like to thank the Fulton Supercomputing Center at BYU for providing the computing resources required for experiments reported here.

References

ABBYY. 2010. ABBYY finereader. http://finereader.abbyy.com.

S. Agarwal, S. Godbole, D. Punjani, and Shourya Roy. 2007. How much noise is too much: A study in automatic text classification. In *Proceedings of the Seventh IEEE Intl. Conf. on Data Mining (ICDM 2007)*, pages 3–12.

Steven M. Beitzel, Eric C. Jensen, and David A. Grossman. 2003. A survey of retrieval strategies for ocr text collections. In *In Proceedings of the Symposium on Document Image Understanding Technologies*.

Michael W. Berry, Murray Brown, and Ben Signer. 2007. 2001 topic annotated Enron email data set.

David M. Blei and John D. Lafferty. 2006. Dynamic topic models. In *Proceedings of the 23rd Intl. Conf. on Machine Learning (ICML 2006)*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Horst Bunke. 2003. Recognition of cursive roman handwriting- past, present and future. In *7th International Conference on Document Analysis and Recognition (ICDAR 2003)*, volume 1, pages 448–459.

Jonathan Chang, Jordan Boyd-Graber, Sean Gerrish, Chong Wang, and David Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Advances in Neural Information Processing Systems 22*, pages 288–296.

Faisal Farooq, Anurag Bhardwaj, and Venu Govindaraju. 2009. Using topic models for OCR correction. *Intl. Journal on Document Analysis and Recognition (IJDAR)*, 12(3), September.

Google, Inc. 2010. Tesseract. http://code.google.com/p/tesseract-ocr.

Thomas L. Griffiths, Mark Steyvers, David M. Blei, and Joshua B. Tenenbaum. 2005. Integrating topics and syntax. In *In Advances in Neural Information Processing Systems 17*, pages 537–544. MIT Press.

Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of Classification*, 2(1):193–218, December.

David Reed Jordan. 1945. Daily battle communiques, 1944-1945. Harold B. Lee Library, L. Tom Perry Special Collections, MSS 2766.

Ken Lang. 1995. NewsWeeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339.

D. Lewis. 1997. Reuters-21578 text categorization test collection. *http://www.research.att.com/˜lewis*.

Tao Liu, Shengping Liu, Zheng Chen, and Wei-Ying Ma. 2003. An evaluation on feature selection for text clustering. In *Proceedings of the Twentieth Intl. Conf. on Machine Learning (ICML 2003)*, August.

Daniel Lopresti. 2008. Optical character recognition errors and their effects on natural language processing. In *Proceedings of the second workshop on Analytics for noisy unstructured text data (AND 2008)*, pages 9–16.

William B. Lund and Eric. K Ringger. 2009. Improving optical character recognition through efficient multiple system alignment. In *Proceedings of the Joint Conf. on Digital Libraries (JCDL'09)*, June.

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

Marina Meilă. 2007. Comparing clusterings—an information based distance. *Journal of Multivariate Analysis*, 98(5):873–895.

David Mimno and Andrew Mccallum. 2007. Organizing the OCA: learning faceted subjects from a library of digital books. In *Proceedings of the Joint Conf. on Digital Libraries (JCDL'07)*, pages 376–385.

Cosmin Munteanu, Ronald Baecker, Gerald Penn, Elaine Toms, and David James. 2006. The effect of speech recognition accuracy rates on the usefulness and usability of webcast archives. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 493–502.

David J. Newmann and Sharon Block. 2006. Probabilistic topic decomposition of an eighteenth-century american newspaper. *J. Am. Soc. Inf. Sci. Technol.*, 57(6):753–767, February.

Nuance Communications, Inc. 2010. OmniPage Pro. http://www.nuance.com/imaging/products/omnipage.asp.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Language Learning (EMNLP-CoNLL 2007)*.

Kazem Taghva, Julie Borsack, and Allen Condit. 1994. Results of applying probabilistic ir to ocr text. In *in Proc. 17th Intl. ACM/SIGIR Conf. on Research and Development in Information Retrieval*, pages 202–211.

Kazem Taghva, Tom Nartker, Julie Borsack, Steve Lumos, Allen Condit, and Ron Young. 2001. Evaluating text categorization in the presence of ocr errors. In *In Proc. IS&T/SPIE 2001 Intl. Symp. on Electronic Imaging Science and Technology*, pages 68–74. SPIE.

Daniel Walker and Eric Ringger. 2008. Model-based document clustering with a collapsed gibbs sampler. In *Proceedings of the 14th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (KDD 2008)*.

Daniel Walker and Erik K. Ringger. 2010. Top N per document: Fast and effective unsupervised feature selection for document clustering. Technical Report 6, Brigham Young University. http://nlp.cs.byu.edu/techreports/BYUNLP-TR6.pdf.

Hanna Wallach, David Mimno, and Andrew McCallum. 2009a. Rethinking LDA: Why priors matter. In *Advances in Neural Information Processing Systems 22*, pages 1973–1981.

Hanna M. Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. 2009b. Evaluation methods for topic models. In *Proceedings of the 26th Annual Intl. Conf. on Machine Learning (ICML 2009)*, pages 1105–1112.

Xuerui Wang and Andrew McCallum. 2006. Topics over time: A non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (KDD 2006)*.

Michael L. Wick, Michael G. Ross, and Erik G. Learned-Miller. 2007. Context-sensitive error correction: Using topic models to improve OCR. In *Proceedings of the Ninth Intl. Conf. on Document Analysis and Recognition (ICDAR 2007)*, pages 1168–1172.

Translingual Document Representations from Discriminative Projections

John C. Platt **Kristina Toutanova** **Wen-tau Yih**
Microsoft Research
1 Microsoft Way
Redmond, WA 98005, USA
{jplatt,kristout,scottyih}@microsoft.com

Abstract

Representing documents by vectors that are independent of language enhances machine translation and multilingual text categorization. We use discriminative training to create a projection of documents from multiple languages into a single translingual vector space. We explore two variants to create these projections: Oriented Principal Component Analysis (OPCA) and Coupled Probabilistic Latent Semantic Analysis (CPLSA). Both of these variants start with a basic model of documents (PCA and PLSA). Each model is then made discriminative by encouraging comparable document pairs to have similar vector representations. We evaluate these algorithms on two tasks: parallel document retrieval for Wikipedia and Europarl documents, and cross-lingual text classification on Reuters. The two discriminative variants, OPCA and CPLSA, significantly outperform their corresponding baselines. The largest differences in performance are observed on the task of retrieval when the documents are only comparable and not parallel. The OPCA method is shown to perform best.

1 Introduction

Given the growth of multiple languages on the Internet, Natural Language Processing must operate on dozens of languages. It is becoming critical that computers reach high performance on the following two tasks:

- **Comparable and parallel document retrieval** — Cross-language information retrieval and text categorization have become important with the growth of the Web (Oard and Diekema, 1998). In addition, machine translation (MT) systems can be improved by training on sentences extracted from parallel or comparable documents mined from the Web (Munteanu and Marcu, 2005). Comparable documents can also be used for learning word-level translation lexicons (Fung and Yee, 1998; Rapp, 1999).
- **Cross-language text categorization** — Applications of text categorization, such as sentiment classification (Pang et al., 2002), are now required to run on multiple languages. Categorization is usually trained on the language of the developer: it needs to be easily extended to other languages.

There are two broad approaches to comparable document retrieval and cross-language text categorization. One approach is to translate queries or a training set from different languages into a single target language. Standard monolingual retrieval and classification algorithms can then be applied in the target language.

Alternatively, a cross-language system can project a bag-of-words vector into a translingual lower-dimensional vector space. Ideally, vectors in this space represent the semantics of a document, independent of the language.

The advantage of pre-translation is that MT systems tend to preserve the meaning of documents. However, MT can be very slow (more than 1 second per document), preventing its use on large training sets. When full MT is not practical, a fast word-by-word translation model can be used instead, (Ballesteros and Croft, 1996) but may be less accurate.

Conversely, applying a projection into a low-dimensional space is quick. Linear projection algorithms use matrix-sparse vector multiplication, which can be easily parallelized. However, as seen in section 3, the accuracies of previous projection

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 251–261,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

techniques are not as high as machine translation.

This paper presents two techniques: Oriented PCA and Coupled PLSA. These techniques retain the high speed of projection, while approaching or exceeding the quality level of word glossing. We improve the quality of the projections by the use of discriminative training: we minimize the difference between comparable documents in the projected vector space. Oriented PCA minimizes the difference by modifying the eigensystem of PCA (Diamantaras and Kung, 1996), while Coupled PLSA uses posterior regularization (Graca et al., 2008; Ganchev et al., 2009) on the topic assignments of the comparable documents.

1.1 Previous work

There has been extensive work in projecting monolingual documents into a vector space. The initial algorithm for projecting documents was Latent Semantic Analysis (LSA), which modeled bag-of-word vectors as low-rank Gaussians (Deerwester et al., 1990). Subsequent projection algorithms were based on generative models of individual terms in the documents, including Probabilistic Latent Semantic Analysis (PLSA) (Hofmann, 1999) and Latent Dirichlet Allocation (LDA) (Blei et al., 2003).

Work on cross-lingual projections followed a similar pattern of moving from Gaussian models to term-wise generative models. Cross-language Latent Semantic Indexing (CL-LSI) (Dumais et al., 1997) applied LSA to concatenated comparable documents from multiple languages. Similarly, Polylingual Topic Models (PLTM) (Mimno et al., 2009) generalized LDA to tuples of documents from multiple languages. The experiments in section 3 use CL-LSI and an algorithm similar to PLTM as benchmarks.

The closest previous work to this paper is the use of Canonical Correlation Analysis (CCA) to find projections for multiple languages whose results are maximally correlated with each other (Vinokourov et al., 2003).

PLSA-, LDA-, and CCA-based cross-lingual models have also been trained without the use of parallel or comparable documents, using only knowledge from a translation dictionary to achieve sharing of topics across languages (Haghighi et al., 2008; Jagarlamudi and Daumé, 2010; Zhang et al., 2010). Such work is complementary to ours and can be used to extend the models to domains lacking parallel documents.

Outside of NLP, researchers have designed algorithms to find discriminative projections. We build on the Oriented Principal Component Analysis (OPCA) algorithm (Diamantaras and Kung, 1996), which finds projections that maximize a signal-to-noise ratio (as defined by the user). OPCA has been used to create discriminative features for audio fingerprinting (Burges et al., 2003).

1.2 Structure of paper

This paper now presents two algorithms for translingual document projection (in section 2): OPCA and Coupled PLSA (CPLSA). To explain OPCA, we first review CL-LSI in section 2.1, then discuss the details of OPCA (section 2.2), and compare it to CCA (section 2.3). To explain CPLSA, we first introduce Joint PLSA (JPLSA), analogous to CL-LSI, in section 2.4, and then describe the details of CPLSA (section 2.5).

We have evaluated these algorithms on two different tasks: comparable document retrieval (section 3.2) and cross-language text categorization (section 3.3). We discuss the findings of the evaluations and extensions to the algorithms in section 4.

2 Algorithms for translingual document projection

2.1 Cross-language Latent Semantic Indexing

Cross-language Latent Semantic Indexing (CL-LSI) is Latent Semantic Analysis (LSA) applied to multiple languages. First, we review the mathematics of LSA.

LSA models an $n \times k$ document-term matrix $\mathbf{D}$, where n is the number of documents and k is the number of terms. The model of the document-term matrix is a low-rank Gaussian. Originally, LSA was presented as performing a Singular Value Decomposition (Deerwester et al., 1990), but here we present it as eigendecomposition, to clarify its relationship with OPCA.

LSA first computes the correlation matrix between terms:

$$\mathbf{C} = \mathbf{D}^T\mathbf{D}. \tag{1}$$

The Rayleigh quotient for a vector $\vec{v}$ with the matrix $\mathbf{C}$ is

$$\frac{\vec{v}^T \mathbf{C} \vec{v}}{\vec{v}^T \vec{v}}, \tag{2}$$

and is equal to the variance of the data projected using the vector $\vec{v}$, normalized by the length of $\vec{v}$, if $\mathbf{D}$ has columns that are zero mean. Good projections retain a large amount of variance. LSA maximizes the Rayleigh ratio by taking its derivative against $\vec{v}$ and setting it to zero. This yields a set of projections that are eigenvectors of $\mathbf{C}$,

$$\mathbf{C}\vec{v}_j = \lambda_j \vec{v}_j, \tag{3}$$

where λ_j is the jth-largest eigenvalue. Each eigenvalue is also the variance of the data when projected by the corresponding eigenvector $\vec{v}_j$. LSA simply uses top d eigenvectors as projections.

LSA is very similar to Principal Components Analysis (PCA). The only difference is that the correlation matrix $\mathbf{C}$ is used, instead of the covariance matrix. In practice, the document-term matrix $\mathbf{D}$ is sparse, so the column means are close to zero, and the correlation matrix is close to the covariance matrix.

There are a number of methods to form the document-term matrix $\mathbf{D}$. One method that works well in practice is to compute the log(tf)-idf weighting: (Dumais, 1990; Wild et al., 2005)

$$D_{ij} = \log_2(f_{ij} + 1) \log_2(n/d_j), \tag{4}$$

where f_{ij} is the number of times term j occurs in document i, n is the total number of documents, and d_j is the total number of documents that contain term j. Applying a logarthm to the term counts makes the distribution of matrix entries approach Gaussian, which makes the LSA model more valid.

Cross-language LSI is an application of LSA where each row of $\mathbf{D}$ is formed by concatenating comparable or parallel documents in multiple languages. If a single term occurs in multiple languages, the term only has one slot in the concatenation, and the term count accumulates for all languages. Such terms could be proper nouns, such as "Smith" or "Merkel".

In general, the elements of $\mathbf{D}$ are computed via

$$D_{ij} = \log_2\left(\sum_m f_{ij}^m + 1\right) \log_2(n/d_j), \tag{5}$$

where f_{ij}^m is the number of times term j occurs in document i in language m. Here, d_j is the number of documents term j appears in, and n is the total number of documents across all languages.

Because CL-LSI is simply LSA applied to concatenated documents, it models terms in document vectors jointly across languages as a single low-rank Gaussian.

2.2 Oriented Principal Component Analysis

The limitations of CL-LSI can be illustrated by considering Oriented Principal Components Analysis (OPCA), a generalization of PCA. A user of OPCA computes a signal covariance matrix $\mathbf{S}$ and a noise covariance matrix $\mathbf{N}$. OPCA projections $\vec{v}_j$ maximize the ratio of the variance of the signal projected by $\vec{v}_j$ to the variance of the noise projected by $\vec{v}_j$. This signal-to-noise ratio is the generalized Rayleigh quotient: (Diamantaras and Kung, 1996)

$$\frac{\vec{v}^T \mathbf{S} \vec{v}}{\vec{v}^T \mathbf{N} \vec{v}}. \tag{6}$$

Taking the derivative of the Rayleigh quotient with respect to the projections $\vec{v}$ and setting it to zero yields the generalized eigenproblem

$$\mathbf{S}\vec{v}_j = \lambda_j \mathbf{N} \vec{v}_j. \tag{7}$$

This eigenproblem has no local minima, and can be solved with commonly available parallel code.

PCA is a specialization of OPCA, where the noise covariance matrix is assumed to be the identity (i.e., uncorrelated noise). PCA projections maximize the signal-to-noise ratio where the signal is the empirical covariance of the data, and the noise is spherical white noise. PCA projections are not truly appropriate for forming multilingual document projections.

Instead, we want multilingual document projections to maximize the projected covariance of document vectors across all languages, while simultaneously minimizing the projected distance between comparable documents (see Figure 1). OPCA gives us a framework for finding such discriminative projections. The covariance matrix for all documents is the signal covariance in OPCA, and captures the meaning of documents across all languages. The projection of this covariance matrix should be maximized. The covariance matrix formed from differences between comparable documents is the noise covariance in OPCA: we wish to minimize the latter covariance, to make the projection language-independent.

Specifically, we create the weighted document-term matrix $\mathbf{D}_m$ for each language:

$$D_{ij,m} = \log_2(f_{ij}^m + 1)log_2(n/d_j). \tag{8}$$

We then derive a signal covariance matrix over all languages:

$$\mathbf{S} = \sum_m \mathbf{D}_m^T \mathbf{D}_m / n - \vec{\mu}_m^T \vec{\mu}_m, \tag{9}$$

where $\vec{\mu}_m$ is the mean of each $\mathbf{D}_m$ over its columns, and a noise covariance matrix,

$$\mathbf{N} = \sum_m (\mathbf{D}_m - \mathbf{D})^T (\mathbf{D}_m - \mathbf{D})/n + \gamma \mathbf{I}, \tag{10}$$

where $\mathbf{D}$ is the mean across all languages of the document-term matrix,

$$\mathbf{D} = \frac{1}{M} \sum_m \mathbf{D}_m, \tag{11}$$

and M is the number of languages. Applying equation (7) to these matrices and taking the top generalized eigenvectors yields the projection matrix for OPCA.

Note the regularization term of $\gamma \mathbf{I}$ in equation (10). The empirical sample of comparable documents may not cover the entire space of translation noise the system will encounter in the test set. For safety, we add a regularizer that prevents the variance of a term from getting too small. We tuned γ on the development sets in section 3.2: for log(tf)-idf weighted vectors, $C = 0.1$ works well for the data sets and dimensionalities that we tried. We use $C = 0.1$ for all final tests.

2.3 Canonical Correlation Analysis

Canonical Correlation Analysis (CCA) is a technique that is related to OPCA. CCA was kernelized and applied to creating cross-language document models by (Vinokourov et al., 2003). In CCA, a linear projection is found for each language, such that the projections of the corpus from each language are maximally correlated with each other. Similar to OPCA, this linear projection can be found by finding the top generalized eigenvectors of the system (7), where $\mathbf{S}$ is now a matrix of cross-correlations that the projection maximizes,

$$\mathbf{S} = \begin{bmatrix} 0 & \mathbf{C}_{12} \\ \mathbf{C}_{21} & 0 \end{bmatrix}, \tag{12}$$

and $\mathbf{N}$ is a matrix of autocorrelations that the projection minimizes,

$$\mathbf{N} = \begin{bmatrix} \mathbf{C}_{11} + \gamma I & 0 \\ 0 & \mathbf{C}_{22} + \gamma I \end{bmatrix}. \tag{13}$$

Here, C_{ij} is the (cross-)covariance matrix, with dimension equal to the vocabulary size, that is computed between the document vectors for languages i and j. Analogous to OPCA, γ is a regularization term, set by optimizing performance on a validation set. Like OPCA, these matrices can be generalized to more than two languages. Unlike OPCA, CCA finds projections that maximize the cross-covariance between the projected vectors, instead of minimizing Euclidean distance.[1]

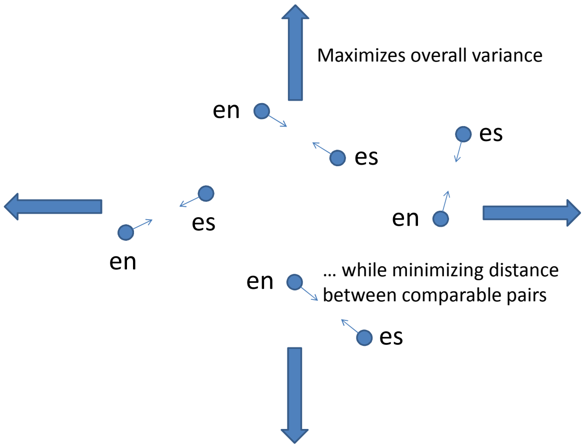


Figure 1: OPCA finds a projection that maximizes the variance of all documents, while minimizing distance between comparable documents

By definition, CCA cannot take advantage of the information that same term occurs simultaneously in comparable documents. As shown in section 3, this

[1] Note that the eigenvectors have length equal to the sum of the length of the vocabularies of each language. The projections for each language are created by splitting the eigenvectors into sections, each with length equal to the vocabulary size for each language.

information is useful and helps OPCA perform better then CCA. In addition, CCA encourages comparable documents to be projected to vectors that are mutually linearly predictable. This is not the same OPCA's projected vectors that have low Euclidean distance: the latter may be preferred by algorithms that consume the projections.

2.4 Cross-language Topic Models

We now turn to a baseline generative model that is analogous to CL-LSI. Our baseline joint PLSA model (JPLSA) is closely related to the poly-lingual LDA model of (Mimno et al., 2009). The graphical model for JPLSA is shown at the top in Figure 2. We describe the model for two languages, but it is straightforward to generalize to more than two languages, as in (Mimno et al., 2009).

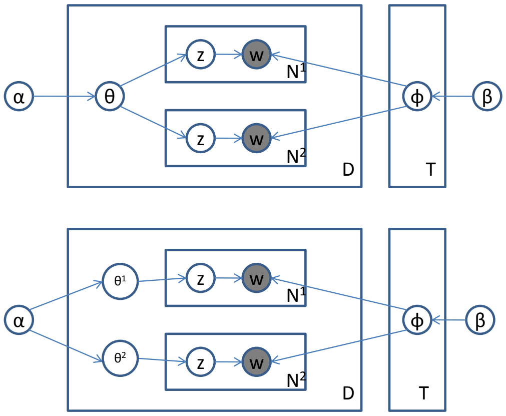


Figure 2: Graphical models for JPLSA (top) and CPLSA (bottom)

The model sees documents d_i as sequences of words $w_1, w_2, \ldots, w_{n_i}$ from a vocabulary V. There are T cross-language topics, each of which has a distribution ϕ_t over words in V. In the case of models for two languages, we define the vocabulary V to contain word types from both languages. In this way, each topic is shared across languages.

Each topic-specific distribution ϕ_t, for $t = 1 \ldots T$, is drawn from a symmetric Dirichlet prior with concentration parameter β. Given the topic-specific word distributions, the generative process for a corpus of paired documents $[d_i^1, d_i^2]$ in two languages L_1 and L_2 is described in the next paragraph.

For each pair of documents, pick a distribution over topics θ_i, from a symmetric Dirichlet prior with concentration parameter α. Then generate the documents d_i^1 and d_i^2 in turn. Each word token in each document is generated independently by first picking a topic z from a multinomial distribution with parameter θ_i ($\textsc{Multi}(\theta_i)$), and then generating the word token from the topic-specific word distribution for the chosen topic $\textsc{Multi}(\phi_z)$.

The probability of a document pair $[d^1, d^2]$ with words $[w_1^1, w_2^1, \ldots, w_{n_1}^1]$, $[w_1^2, w_2^2, \ldots, w_{n_2}^2]$, topic assignments $[z_1^1, \ldots, z_{n_1}^1]$, $[z_1^2, \ldots, z_{n_2}^2]$, and a common topic vector θ is given by:

$$P(\theta|\alpha) \prod_{j=1}^{n_1} P(z_j^1|\theta) P(w_j^1|\phi_{z_j^1}) \prod_{j=1}^{n_2} P(z_j^2|\theta) P(w_j^2|\phi_{z_j^2})$$

The difference between the JPLSA model and the poly-lingual topic model of (Mimno et al., 2009) is that we merge the vocabularies in the two languages and learn topic-specific word distributions over these merged vocabularies, instead of having pairs of topic-specific word distributions, one for each language, like in (Mimno et al., 2009). Thus our model is more similar to the CL-LSI model, because it can be seen as viewing a pair of documents in two languages as one bigger document containing the words in both documents.

Another difference between our model and the poly-lingual LDA model of (Mimno et al., 2009) is that we use maximum aposteriori (MAP) instead of Bayesian inference. Recently, MAP inference was shown to perform comparably to the best inference method for LDA (Asuncion et al., 2009), if the hyper-parameters are chosen optimally for the inference method. Our initial experiments with Bayesian versus MAP inference for parallel document retrieval using JPLSA confirmed this result. In practice our baseline model outperforms poly-lingual LDA as mentioned in our experiments.

2.5 Coupled Probabilistic Latent Semantic Analysis

The JPLSA model assumes that a pair of translated or comparable documents have a common topic distribution θ. JPLSA fits its parameters to optimize the probability of the data, given this assumption.

For the task of comparable document retrieval, we want our topic model to assign similar topic distributions θ to a pair of corresponding documents. But this is not exactly what the JPLSA model is doing. Instead, it derives a common topic vector θ which explains the union of all tokens in the English and foreign documents, instead of making sure that the best topic assignment for the English document is close to the best topic assignment of the foreign document. This difference becomes especially apparent when corresponding documents have different lengths. In this case, the model will tend to derive a topic vector θ which explains the longer document best, making the sum of the two documents' log-likelihoods higher. Modeling the shorter document's best topic carries little weight.

Modeling both documents equally is what Coupled PLSA (CPLSA) is designed to do. The graphical model for CPLSA is shown at the bottom of Figure 2. In this figure, the topic vectors of a pair of documents in two languages are shown completely independent. We use the log-likelihood according to this model, but also add a regularization term, which tries to make the topic assignments of corresponding documents close. In particular, we use posterior regularization (Graca et al., 2008; Ganchev et al., 2009) to place linear constraints on the expectations of topic assignments to two corresponding documents.

For two linked documents d_1 and d_2, we would like our model to be such that the expected fraction of tokens in d_1 that get assigned topic t is approximately the same as the expected fraction of tokens in d_2 that get assigned the same topic t, for each topic $t = 1 \dots T$. This is exactly what we need to make each pair of corresponding documents close.

Let $\mathbf{z^1}$ and $\mathbf{z^2}$ denote vectors of topic assignments to the tokens in document d^1 and d^2, respectively. Their dimensionality is equal to the lengths of the two documents, n_1 and n_2. We define a space of posterior distributions Q over hidden topic assignments to the tokens in d^1 and d^2, that has the desired property: the expected fraction of each topic is approximately equal in d^1 and d^2. We can formulate this constrained space Q as follows:

$$Q = \{q_1(\mathbf{z^1}), q_2(\mathbf{z^2})\}$$

such that

$$\mathbf{E}_{q_1}[\frac{\sum_{j=1}^{n_1} \mathbf{1}(z_j^1 = t)}{n_1}] - \mathbf{E}_{q_2}[\frac{\sum_{j=1}^{n_2} \mathbf{1}(z_j^2 = t)}{n_2}] \le \epsilon_t$$

$$\mathbf{E}_{q_2}[\frac{\sum_{j=1}^{n_2} \mathbf{1}(z_j^2 = t)}{n_2}] - \mathbf{E}_{q_1}[\frac{\sum_{j=1}^{n_1} \mathbf{1}(z_j^1 = t)}{n_1}] \le \epsilon_t$$

We then formulate an objective function that maximizes the log-likelihood of the data while simultaneously minimizing the KL-divergence between the desired distribution set Q and the posterior distribution according to the model: $P(\mathbf{z_1}|d_1, \theta_1, \phi)$ and $P(\mathbf{z_2}|d_2, \theta_2, \phi)$.

The objective function for a single document pair is as follows:

$$\begin{aligned}&\log P(d_1|\theta_1, \phi) + \log P(d_2|\theta_2, \phi)\\ &\qquad -\mathbf{KL}(Q||P(\mathbf{z_1}|d_1, \theta_1, \phi), P(\mathbf{z_2}|d_2, \theta_2, \phi))\\ &\qquad -||\epsilon||\end{aligned}$$

The final corpus-wide objective is summed over document-pairs, and also contains terms for the probabilities of the parameters θ and ϕ given the Dirichlet priors. The norm of ϵ is minimized, which makes the expected proportions of topics in two documents as close as possible.

Following (Ganchev et al., 2009), we fit the parameters by an EM-like algorithm, where for each document pair, after finding the posterior distribution of the hidden variables, we find the KL-projection of this posterior onto the constraint set, and take expected counts with respect to this projection; these expected counts are used in the M-step. The projection is found using a simple projected gradient algorithm.[2]

For both the baseline JPLSA and the CPLSA models, we performed learning through MAP inference using EM (with a projection step for CPLSA). We did up to 500 iterations for each model, and did early stopping based on task performance on the development set. The JPLSA model required more iterations before reaching its peak accuracy, tending to require around 300 to 450 iterations for convergence. CPLSA required fewer iterations, but each iteration was slower due to the projection step.

[2] We initialized the models deterministically by assigning each word to exactly one topic to begin with, such that all topics have roughly the same number of words. Words were sorted by frequency and thus words of similar frequency are more likely to be assigned to the same topic.This initialization method outperformed random initialization and we use it for all models.

All models use $\alpha = 1.1$ and $\beta = 1.01$ for the values of the concentration parameters. We found that the performance of the models was not very sensitive to these values, in the region that we tested ($\alpha, \beta \in [1.001, 1.1]$). Higher hyper-parameter values resulted in faster convergence, but the final performance was similar across these different values.

3 Experimental validation

We test the proposed discriminative projections versus more established cross-language models on the two tasks described in the introduction: retrieving comparable documents from a corpus, and training a classifier in one language and using it in another. We measure accuracy on a test set, and also examine the sensitivity to dimensionality of the projection on development sets.

3.1 Speed of training and evaluation

We first test the speed of the various algorithms discussed in this paper, compared to a full machine translation system. When finding document projections, CL-LSI, OPCA, CCA, JPLSA, and CPLSA are equally fast: they perform a matrix multiplication and require $O(nk)$ operations, where n is the number of distinct words in the documents and k is the dimensionality of the projection.[3] A single CPU core can read the indexed documents into memory and take logarithms at 216K words per second. Projecting into a 2000-dimensional space operates at 41K words per second. Translating word-by-word operates at 274K words per second. In contrast, machine translation processes 50 words per second, approximately 3 orders of magnitude slower.

Total training time for OPCA on 43,380 pairs of comparable documents was 90 minutes, running on an 8-core CPU for 2000 dimensions. On the same corpus, JPLSA requires 31 minutes per iteration and CPLSA requires 377 minutes per iteration. CPLSA requires a factor of five times fewer iterations: overall, it is twice as slow as JPLSA.

3.2 Retrieval of comparable documents

In comparable document retrieval, a query is a document in one language, which is compared to a corpus of documents in another language. By mapping all documents into the same vector space, the comparison is a vector comparison. For our experiments with CL-LSI, OPCA, and CCA, we use cosine similarity between vectors to rank the documents.

For the JPLSA and CPLSA models, we map the documents to corresponding topic vectors θ, and compute distance between these probability vectors. The mapping to topic vectors requires EM iterations, or folding-in (Hofmann, 1999). We found that performing a single EM iteration resulted in best performance so we used this for all models. For computing distance we used the L1-norm of the difference, which worked a bit better than the Jensen-Shannon divergence between the topic vectors used in (Mimno et al., 2009).

We test all algorithms on the Europarl data set of documents in English and Spanish, and a set of Wikipedia articles in English and Spanish that contain interlanguage links between them (i.e., articles that the Wikipedia community have identified as comparable across languages).

For the Europarl data set, we use 52,685 documents as training, 11,933 documents as a development set, and 18,415 documents as a final test set. Documents are defined as speeches by a single speaker, as in (Mimno et al., 2009).[4] For the Wikipedia set, we use 43,380 training documents, 8,675 development documents, and 8,675 final test set documents.

For both corpora, the terms are extracted by word-breaking all documents, removing the top 50 most frequent terms and keeping the next 20,000 most frequent terms. No stemming or folding is applied.

We assess performance by testing each document in English against all possible documents in Spanish, and *vice versa*. We measure the Top-1 accuracy (i.e., whether the true comparable is the closest in the test set), and the Mean Reciprocal Rank of the true comparable, and report the average performance over the two retrieval directions. Ties are counted as errors.

We tuned the dimensionality of the projections on the development set, as shown in Figures 3 and 4.

[3] For JPLSA and CPLSA this is the case only when performing a single EM iteration at test time, which we found to perform best.

[4] The training section contains documents from the years 96 through 99 and the year 02; the dev section contains documents from 01, and the test section contains documents from 00 plus the first 9 months of 03.

We chose the best dimension on the development set for each algorithm, and used it on the final test set. The regularization γ was tuned for CCA: $\gamma = 10$ for Europarl, and $\gamma = 3$ for Wikipedia.

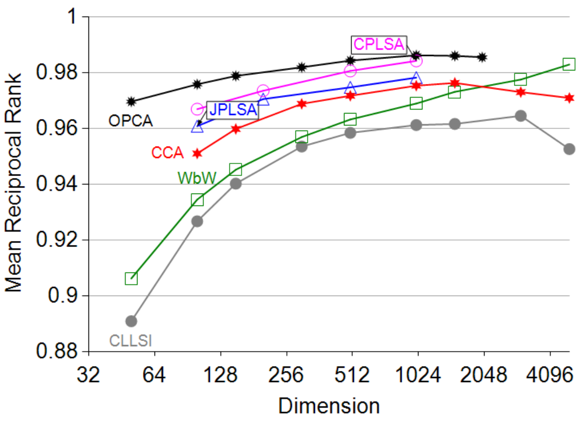


Figure 3: Mean reciprocal rank versus dimension for Europarl

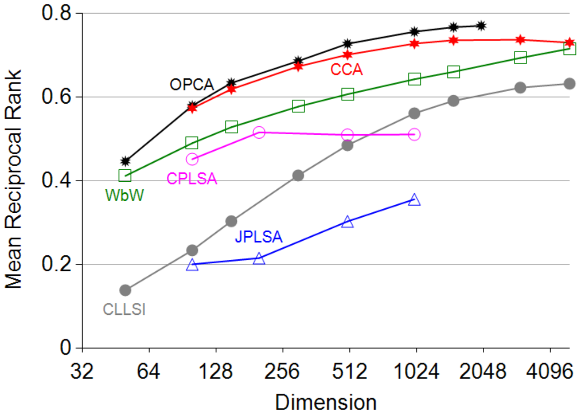


Figure 4: Mean reciprocal rank versus dimension for Wikipedia

In the two figures, we evaluate the five projection methods, as well as a word-by-word translation method (denoted by WbW in the graphs). Here "word-by-word" refers to using cosine distance after applying a word-by-word translation model to the Spanish documents.

The word-by-word translation model was trained on the Europarl training set, using the WDHMM model (He, 2007), which performs similarly to IBM Model 4. The probability matrix of generating English words from Spanish words was multiplied by each document's log(tf)-idf vector to produce a translated document vector. We found that multiplying the probability matrix to the log(tf)-idf vector was more accurate on the development set than multiplying the tf vector directly. This vector was either tested as-is, or mapped through LSA learned from the English training set of the corpus. In the figures, the dimensionality of WbW translation refers to the dimensionality of monolingual LSA.

The overall ordering of the six models is different for the Europarl and Wikipedia development datasets. The discriminative models outperform the corresponding generative ones (OPCA vs CL-LSI) and (CPLSA vs JPLSA) for both datasets, and OPCA performs best overall, dominating the best fast-translation based model, as well as the other projection methods, including CCA.

On Europarl, JPLSA and CPLSA outperform CL-LSI, with the best dimension or JPLSA also slightly outperforming the best setting for the word-by-word translation model, whereas on Wikipedia the PLSA-based models are significantly worse than the other models.

The results on the final test set, evaluating each model using its best dimensionality setting, confirm the trends observed on the development set. The final results are shown in Tables 1 and 2. For these experiments, we use the unpaired t-test with Bonferroni correction to determine the smallest set of algorithms that have statistically significantly better accuracy than the rest. The p-value threshold for significance is chosen to be 0.05. The accuracies for these significantly superior algorithms are shown in boldface.

For Wikipedia and Europarl, we include an additional baseline model,"Untranslated": this refers to applying cosine distance to both the Spanish and English documents directly (since they share some vocabulary terms). For Wikipedia, comparable documents seem to share many common terms, so cosine distance between untranslated documents is a reasonable benchmark.

From the final Europarl results we can see that the best models can learn to retrieve parallel documents from the narrow Europarl domain very well. All dimensionality reduction methods can learn from cleanly parallel data, but discriminative training can bring additional error reduction.

In previously reported work, (Mimno et al., 2009) evaluate parallel document retrieval using PLTM on Europarl speeches in English and Spanish, using training and test sets of size similar to ours. They report an accuracy of 81.2% when restricting to test documents of length at least 100 and using 50 topics. JPLSA with 50 topics obtains accuracy of 98.9% for documents of that length.

The final Wikipedia results are also similar to the the development set results. The problem setting for Wikipedia is different, because corresponding documents linked in Wikipedia may have widely varying degrees of parallelism. While most linked documents share some main topics, they could cover different numbers of sub-topics at varying depths. Thus the training data of linked documents is noisy, which makes it hard for projection methods to learn. The word-by-word translation model in this setting is trained on clean, but out-of-domain parallel data (Europarl), so it has the disadvantage that it may not have a good coverage of the vocabulary; however, it is not able to make use of the Wikipedia training data since it requires sentence-aligned translations. We find it encouraging that the best projection method OPCA outperformed word-by-word translation. This means that OPCA is able to uncover topic correspondence given only comparable document pairs, and to learn well in this noisy setting.

The PLSA-based models fare worse on Wikipedia document retrieval. CPLSA outperforms JPLSA more strongly, but both are worse than CL-LSI and even the Untranslated baseline. We think this is partly explained by the diverse vocabulary in the heterogenous Wikipedia collection. All other models use log(tf)-idf weighting, which automatically assigns importance weights to terms, whereas the topic models use word counts. This weighting is very useful for Wikipedia. For example, if we apply the untranslated matching using raw word counts, the MRR is 0.1024 on the test set, compared to 0.5383 for log(tf)-idf. We hypothesize that using a hierarchical topic model that automatically learns about more general and more topic-specific words would be helpful in this case. It is also possible that PLSA-based models require cleaner data to learn well.

The overall conclusion is that OPCA outperformed all other document retrieval methods we tested, including fast machine translation of documents. Additionally, both discriminative projection methods outperformed their generative counterparts.

Algorithm	Dimension	Accuracy	MRR
OPCA	1000	**0.9742**	0.9806
CPLSA	1000	**0.9716**	0.9782
Word-by-word	N/A	**0.9707**	0.9779
Word-by-word	5000	**0.9706**	0.9778
JPLSA	1000	0.9645	0.9726
CCA	1500	0.9613	0.9705
CL-LSI	3000	0.9457	0.9595
Untranslated	N/A	0.1595	0.2564

Table 1: Test results for comparable document retrieval in Europarl. Boldface indicates statistically significant superior results.

Algorithm	Dimension	Accuracy	MRR
OPCA	2000	**0.7255**	0.7734
Word-by-word	N/A	0.7033	0.7467
CCA	1500	0.6894	0.7378
Word-by-word	5000	0.6786	0.7236
CL-LSI	5000	0.5302	0.6130
Untranslated	N/A	0.4692	0.5383
CPLSA	200	0.4579	0.5130
JPLSA	1000	0.3322	0.3619

Table 2: Test results for comparable document retrieval in Wikipedia. Boldface indicates statistically significant best result.

3.3 Cross-language text classification

The second task is to train a text categorization system in one language, and test it with documents in another. To evaluate on this task, we use the Multilingual Reuters Collection, defined and provided by (Amini et al., 2009). We test the English/Spanish language pair. The collection has news articles in English and Spanish, each of which has been translated to the other by the Portage translation system (Ueffing et al., 2007).

From the English news corpus, we take 13,131 documents as training, 1,875 documents as development, and 1,875 documents as test. We take the English training documents translated into Spanish as our comparable training data. For testing, we use the entire Spanish news corpus of 12,342 documents, either mapped with cross-lingual projection, or translated by Portage.

The data set was provided by (Amini et al., 2009) as already-processed document vectors, using BM25 weighting. Thus, we only test OPCA, CL-LSI, and related methods: JPLSA and CPLSA require modeling the term counts directly.

The performance on the task is measured by classification accuracy on the six disjoint category labels defined by (Amini et al., 2009). To introduce minimal bias due to the classifier model, we use 1-nearest neighbor on top of the cosine distance between vectors as a classifier. For all of the techniques, we treated the vocabulary in each language as completely separate, using the top 10,000 terms from each language.

Note that no Spanish labeled data is provided for training any of these algorithms: only English and translated English news is labeled. The optimal dimension (and γ for CCA) on the development set was chosen to maximize the accuracy of English classification and translated English-to-Spanish classification.

Algorithm	Dim.	English Accuracy	Spanish Accuracy
Full MT	50	**0.8483**	**0.6484**
OPCA	100	**0.8412**	0.5954
Word-by-word	50	**0.8483**	0.5780
CCA	150	**0.8388**	0.5384
Full MT	N/A	0.8046	0.5323
CL-LSI	150	**0.8401**	0.5105
Word-by-word	N/A	0.8046	0.4481

Table 3: Test results for cross-language text categorization

The test classification accuracy is shown in Table 3. As above, the smallest set of superior algorithms as determined by Bonferroni-corrected t-tests are shown in boldface. The results for MT and word-by-word translation use the log(tf)-idf vector directly for documents that were written in English, and use a Spanish-to-English translated vector if the document was written in Spanish. As in section 3.2, word-by-word translation multiplied each log(tf)-idf vector by the translation probability matrix trained on Europarl.

The tests show that OPCA is better than CCA, CL-LSI, plain word-by-word translation, and even full translation for Spanish documents. However, if we post-process full translation by an LSI model trained on the English training set, full translation is the most accurate. If full translation is time-prohibitive, then OPCA is the best method: it is significantly better than word-by-word translation followed by LSI.

4 Discussion and Extensions

OPCA extends naturally to multiple languages. However, it requires memory and computation time that scales quadratically with the size of the vocabulary. As the number of languages goes up, it may become impractical to perform OPCA directly on a large vocabulary.

Researchers have solved the problem of scaling OPCA by using Distortion Discriminant Analysis (DDA) (Burges et al., 2003). DDA performs OPCA in two stages which avoids the need for solving a very large generalized eigensystem. As future work, DDA could be applied to mapping documents in many languages simultaneously.

Spherical Admixture Models (Reisinger et al., 2010) have recently been proposed that combine an LDA-like hierarchical generative model with the use of tf-idf representations. A similar model could be used for CPLSA: future work will show whether such a model can outperform OPCA.

5 Conclusions

This paper presents two different methods for creating discriminative projections: OPCA and CPLSA. Both of these methods avoid the use of artificial concatenated documents. Instead, they model documents in multiple languages, with the constraint that comparable documents should map to similar locations in the projected space.

When compared to other techniques, OPCA had the highest accuracy while still having a run-time that allowed scaling to large data sets. We therefore recommend the use of OPCA as a pre-processing step for large-scale comparable document retrieval or cross-language text categorization.

References

Massih-Reza Amini, Nicolas Usunier, and Cyril Goutte. 2009. Learning from multiple partially observed views - an application to multilingual text categorization. In *Advances in Neural Information Processing Systems 22 (NIPS 2009)*, pages 28–36.

Arthur Asuncion, Max Welling, Padhraic Smyth, and Yee Whye Teh. 2009. On smoothing and inference for topic models. In *Proceedings of Uncertainty in Artificial Intelligence*, pages 27–34.

Lisa Ballesteros and Bruce Croft. 1996. Dictionary methods for cross-lingual information retrieval. In *Proceedings of the 7th International DEXA Conference on Database and Expert Systems Applications*, pages 791–801.

David M. Blei, Andrew Y. Ng, Michael I. Jordan, and John Lafferty. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Christopher J.C. Burges, John C. Platt, and Soumya Jana. 2003. Distortion discriminant analysis for audio fingerprinting. *IEEE Transactions on Speech and Audio Processing*, 11(3):165–174.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Konstantinos I. Diamantaras and S.Y. Kung. 1996. *Principal Component Neural Networks: Theory and Applications*. Wiley-Interscience.

Susan T. Dumais, Todd A. Letsche, Michael L. Littman, and Thomas K. Landauer. 1997. Automatic cross-language retrieval using latent semantic indexing. In *AAAI-97 Spring Symposium Series: Cross-Language Text and Speech Retrieval*.

Susan T. Dumais. 1990. Enhancing performance in latent semantic indexing (LSI) retrieval. Technical Report TM-ARH-017527, Bellcore.

Pascale Fung and Lo Yuen Yee. 1998. An IR approach for translating new words from nonparallel, comparable texts. In *Proceedings of COLING-ACL*, pages 414–420.

Kuzman Ganchev, Joao Graca, Jennifer Gillenwater, and Ben Taskar. 2009. Posterior regularization for structured latent variable models. Technical Report MS-CIS-09-16, University of Pennsylvania.

Joao Graca, Kuzman Ganchev, and Ben Taskar. 2008. Expectation maximization and posterior constraints. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 569–576. MIT Press, Cambridge, MA.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proc. ACL*, pages 771–779.

Xiaodong He. 2007. Using word-dependent transition models in HMM based word alignment for statistical machine translation. In *ACL 2nd Statistical MT workshop*, pages 80–87.

Thomas Hofmann. 1999. Probabilistic latent semantic analysis. In *Proceedings of Uncertainty in Artificial Intelligence*, pages 289–296.

Jagadeesh Jagarlamudi and Hal Daumé, III. 2010. Extracting multilingual topics from unaligned comparable corpora. In *ECIR*.

David Mimno, Hanna W. Wallach, Jason Naradowsky, David A. Smith, and Andrew McCallum. 2009. Polylingual topic models. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 880–889.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31:477–504.

Douglas W. Oard and Anne R. Diekema. 1998. Cross-language information retrieval. In Martha Williams, editor, *Annual Review of Information Science (ARIST)*, volume 33, pages 223–256.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proc. EMNLP*, pages 79–86.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated English and German corpora. In *Proceedings of the ACL*, pages 519–526.

Joseph Reisinger, Austin Waters, Bryan Silverthorn, and Raymond J. Mooney. 2010. Spherical topic models. In *Proc. ICML*.

Nicola Ueffing, Michel Simard, Samuel Larkin, and J. Howard Johnson. 2007. NRC's PORTAGE system for WMT 2007. In *ACL-2007 2nd Workshop on SMT*, pages 185–188.

Alexei Vinokourov, John Shawe-Taylor, and Nello Cristianini. 2003. Inferring a semantic representation of text via cross-language correlation analysis. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1473–1480, Cambridge, MA. MIT Press.

Fridolin Wild, Christina Stahl, Gerald Stermsek, and Gustaf Neumann. 2005. Parameters driving effectiveness of automated essay scoring with LSA. In *Proceedings 9th Internaional Computer-Assisted Assessment Conference*, pages 485–494.

Duo Zhang, Qiaozhu Mei, and ChengXiang Zhai. 2010. Cross-lingual latent topic extraction. In *Proc. ACL*, pages 1128–1137, Uppsala, Sweden. Association for Computational Linguistics.

Storing the Web in Memory: Space Efficient Language Models with Constant Time Retrieval

David Guthrie
Computer Science Department
University of Sheffield
D.Guthrie@dcs.shef.ac.uk

Mark Hepple
Computer Science Department
University of Sheffield
M.Hepple@dcs.shef.ac.uk

Abstract

We present three novel methods of compactly storing very large n-gram language models. These methods use substantially less space than all known approaches and allow n-gram probabilities or counts to be retrieved in constant time, at speeds comparable to modern language modeling toolkits. Our basic approach generates an *explicit* minimal perfect hash function, that maps all n-grams in a model to distinct integers to enable storage of associated values. Extensions of this approach exploit distributional characteristics of n-gram data to reduce storage costs, including variable length coding of values and the use of *tiered* structures that partition the data for more efficient storage. We apply our approach to storing the full Google Web1T n-gram set and all 1-to-5 grams of the Gigaword newswire corpus. For the 1.5 billion n-grams of Gigaword, for example, we can store full count information at a cost of 1.66 bytes per n-gram (around 30% of the cost when using the current state-of-the-art approach), or quantized counts for 1.41 bytes per n-gram. For applications that are tolerant of a certain class of relatively innocuous errors (where unseen n-grams may be accepted as rare n-grams), we can reduce the latter cost to below 1 byte per n-gram.

1 Introduction

The availability of very large text collections, such as the Gigaword corpus of newswire (Graff, 2003), and the Google Web1T 1-5gram corpus (Brants and Franz, 2006), have made it possible to build models incorporating counts of billions of n-grams. The storage of these language models, however, presents serious problems, given both their size and the need to provide rapid access. A prevalent approach for language model storage is the use of compact trie structures, but these structures do not scale well and require space proportional to both to the number of n-grams and the vocabulary size. Recent advances (Talbot and Brants, 2008; Talbot and Osborne, 2007b) involve the development of Bloom filter based models, which allow a considerable reduction in the space required to store a model, at the cost of allowing some limited extent of false positives when the model is queried with previously unseen n-grams. The aim is to achieve sufficiently compact representation that even very large language models can be stored totally within memory, avoiding the latencies of disk access. These Bloom filter based models exploit the idea that it is not actually necessary to *store* the n-grams of the model, as long as, when queried with an n-gram, the model returns the correct count or probability for it. These techniques allow the storage of language models that no longer depend on the size of the vocabulary, but only on the number of n-grams.

In this paper we give three different models for the efficient storage of language models. The first structure makes use of an explicit perfect hash function that is *minimal* in that it maps n keys to integers in the range 1 to n. We show that by using a minimal perfect hash function and exploiting the distributional characteristics of the data we produce n-gram models that use less space than all know approaches with no reduction in speed. Our two further models achieve even more compact storage while maintaining constant time access by using variable length coding to compress the n-grams values and by using tiered hash structures to parti-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 262–272,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

tion the data into subsets requiring different amounts of storage. This combination of techniques allows us, for example, to represent the *full* count information of the Google Web1T corpus (Brants and Franz, 2006) (where count values range up to 95 billion) at a cost of just 2.47 bytes per n-gram (assuming 8-bit fingerprints, to exclude false positives) and just 1.41 bytes per n-gram if we use 8-bit quantization of counts. These costs are 36% and 57% respectively of the space required by the Bloomier Filter approach of Talbot and Brants (2008). For the Gigaword dataset, we can store full count information at a cost of only 1.66 bytes per n-gram. We report empirical results showing that our approach allows a look-up rate which is comparable to existing modern language modeling toolkits, and much faster than a competitor approach for space-efficient storage. Finally, we propose the use of *variable length fingerprinting* for use in contexts which can tolerate a higher rate of 'less damaging' errors. This move allows, for example, the cost of storing a quantized model to be reduced to 1 byte per n-gram or less.

2 Related Work

A range of *lossy* methods have been proposed, to reduce the storage requirements of LMs by discarding information. Methods include the use of entropy pruning techniques (Stolcke, 1998) or clustering (Jelinek et al., 1990; Goodman and Gao, 2000) to reduce the number of n-grams that must be stored. A key method is *quantization* (Whittaker and Raj, 2001), which reduces the value information stored with n-grams to a limited set of *discrete* alternatives. It works by grouping together the values (probabilities or counts) associated with n-grams into clusters, and replacing the value to be stored for each n-gram with a code identifying its value's cluster. For a scheme with n clusters, codes require $\log_2 n$ bits. A common case is 8-bit quantization, allowing up to 256 distinct 'quantum' values. Different methods of dividing the range of values into clusters have been used, e.g. Whittaker and Raj (2001) used the Lloyd-Max algorithm, whilst Federico and Bertoldi (2006) use the simpler Binning method to quantize probabilities, and show that the LMs so produced out-perform those produced using the Lloyd-Max method on a phrase-based machine translation task. Binning partitions the range of values into regions that are uniformly populated, i.e. producing clusters that contain the same number of unique values. Functionality to perform uniform quantization of this kind is provided as part of various LM toolkits, such as IRSTLM. Some of the empirical storage results reported later in the paper relate to LMs recording n-gram *count* values which have been quantized using this uniform binning approach. In the rest of this section, we turn to look at some of the approaches used for *storing* language models, irrespective of whether lossy methods are first applied to reduce the size of the model.

2.1 Language model storage using Trie structures

A widely used approach for storing language models employs the *trie* data structure (Fredkin, 1960), which compactly represents sequences in the form of a *prefix tree*, where each step down from the root of the tree adds a new element to the sequence represented by the nodes seen so far. Where two sequences share a prefix, that common prefix is jointly represented by a single node within the trie. For language modeling purposes, the steps through the trie correspond to *words* of the vocabulary, although these are in practice usually represented by 24 or 32 bit integers (that have been uniquely assigned to each word). Nodes in the trie corresponding to *complete* n-grams can store other information, e.g. a probability or count value. Most modern language modeling toolkits employ some version of a trie structure for storage, including SRILM (Stolcke, 2002), CMU toolkit (Clarkson and Rosenfeld, 1997), MITLM (Hsu and Glass, 2008), and IRSTLM (Federico and Cettolo, 2007) and implementations exist which are very compact (Germann et al., 2009). An advantage of this structure is that it allows the stored n-grams to be enumerated. However, although this approach achieves a compact of representation of sequences, its memory costs are still such that very large language models require very large storage space, far more than the Bloom filter based methods described shortly, and far more than might be held in memory as a basis for more rapid access. The memory costs of such models have been addressed using compression methods, see Harb et al. (2009), but such extensions of the approach present further obstacles to rapid access.

2.2 Bloom Filter Based Language Models

Recent randomized language models (Talbot and Osborne, 2007b; Talbot and Osborne, 2007a; Talbot and Brants, 2008; Talbot and Talbot, 2008; Talbot, 2009) make use of Bloom filter like structures to map n-grams to their associated probabilities or counts. These methods store language models in relatively little space by not actually keeping the n-gram key in the structure and by allowing a small probability of returning a false positive, i.e. so that for an n-gram that is not in the model, there is a small risk that the model will return some random probability instead of correctly reporting that the n-gram was not found. These structures do not allow enumeration over the n-grams in the model, but for many applications this is not a requirement and their space advantages make them extremely attractive. Two major approaches have been used for storing language models: Bloom Filters and Bloomier Filters. We give an overview of both in what follows.

2.2.1 Bloom Filters

A Bloom filter (Bloom, 1970) is a compact data structure for membership queries, i.e. queries of the form "Is this key in the Set?". This is a weaker structure than a dictionary or hash table which also associates a value with a key. Bloom filters use well below the information theoretic lower bound of space required to actually store the keys and can answer queries in O(1) time. Bloom filters achieve this feat by allowing a small probability of returning a false positive. A Bloom filter stores a set S of n elements in a bit array B of size m. Initially B is set to contain all zeros. To store an item x from S in B we compute k random independent hash functions on x that each return a value in the range $[0 \,..\, m-1]$. These values serve as indices to the bit array B and the bits at those positions are set to 1. We do this for all elements in S, storing to the same bit array. Elements may hash to an index in B that has already been set to 1 and in this case we can think of these elements as "sharing" this bit. To test whether set S contains a key w, we run our k hash functions on w and check if all those locations in B are set to 1. If $w \in S$ then the bloom filter will always declare that w belongs to S, but if $x \notin S$ then the filter can only say with high probability that w is not in S. This error rate depends on the number of k hash functions and the ratio of m/n. For instance with $k = 3$ hash functions and a bit array of size $m = 20n$, we can expect to get a false positive rate of 0.0027.

Talbot and Osborne (2007b) and Talbot and Osborne (2007a) adapt Bloom filters to the requirement of storing values for n-grams by concatenating the key (n-gram) and value to form a single item that is inserted into the filter. Given a quantization scheme allowing values in the range $[1 \,..\, V]$, a quantized value v is stored by inserting into the filter all pairings of the n-gram with values from 1 up to v. To retrieve the value for a given key, we serially probe the filter for pairings of the key with each value from 1 upwards, until the filter returns false. The last value found paired with the key in the filter is the value returned. Talbot and Osborne use a simple logarithmic quantization of counts that produce limited quantized value ranges, where most items will have values that are low in the range, so that the serial look-up process will require quite a low number of steps *on average*. For alternative quantization schemes that involve greater value ranges (e.g. the 256 values of a uniform 8-bit scheme) and/or distribute n-grams more evenly across the quantized values, the average number of look-up steps required will be higher and hence the speed of access per n-gram accordingly lower. In that case also, the requirement of inserting n-grams more than once in the filter (i.e. with values from 1 up to the actual value v being stored) could substantially reduce the space efficiency of the method, especially if low false positive rates are required, e.g. the case $k = 3, m = 20n$ produces a false positive rate of 0.0027, as noted above, but in a situation where 3 key-value items were being stored per n-gram on average, this error rate would in fact require a storage cost of 60 bits per original n-gram.

2.2.2 Bloomier Filters

More recently, Talbot and Brants (2008) have proposed an approach to storing large language models which is based on the *Bloomier Filter* technique of Chazelle et al. (2004). Bloomier Filters generalize the Bloom Filter to allow values for keys to be stored in the filter. To test whether a given key is present in a populated Bloomier filter, we apply k hash functions to the key and use the results as indices for retrieving the data stored at k locations within the filter, similarly to look-up in a Bloom filter. In this case, however, the data retrieved from the filter consists of k *bit vectors*, which are combined with a fingerprint of the key, using *bitwise XOR*, to return the stored value. The risk of false positives is managed by making incorporating a fingerprint of the n-gram, and by making bit vectors longer than the minimum length required to store values. These additional *error bits* have a fairly predictable impact on error rates, i.e. with e error bits, we anticipate the probability of falsely construing an unseen n-gram as being stored in the filter to be 2^{-e}. The algorithm required to correctly populate the Bloomier filter with stored data is complicated, and we shall not consider its details here. Nevertheless, when using v bits to represent values and e bits for error detection, this approach allows a language model to be stored at a cost of is $1.23 \cdot (v + e)$ bits per n-gram.

3 Single Minimal Perfect Hash Ranking Approach

We first describe our basic structure we call Single Minimal Perfect Hash Rank (S-MPHR) that is more compact than that of Talbot and Brants (2008) while still keeping a constant look up time. In the next two sections we describe variations on this model to further reduce the space required while maintaining a constant look up time. The S-MPHR structure can be divided into 3 parts as shown in Figure 1: Stage 1 is a minimal perfect hash function; Stage 2 is a fingerprint and rank array; and Stage 3 is a unique value array. We discuss each stage in turn.

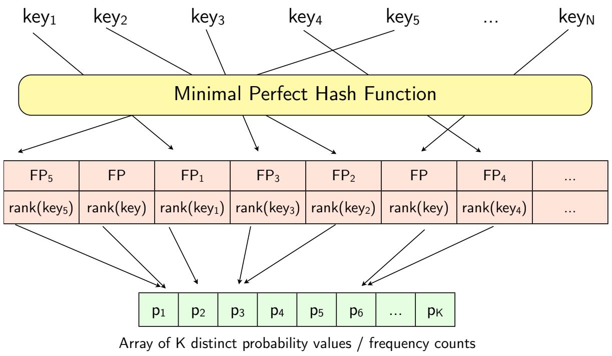


Figure 1: The Single MPHR structure

3.1 Minimal Perfect Hash Function

The first part of the structure is a *minimal* perfect hash function that maps every n-gram in the training data to a distinct integer in the range 0 to $N - 1$, where N is the total number of n-grams to store. We use these integers as indices into the array of Stage 2 of our structure.

We use the *Hash, displace, and compress (CHD)* (Belazzougui et al., 2009) algorithm to generate a minimal perfect hash function that requires 2.07 bits per key and has $O(1)$ access. The algorithm works as follows. Given a set S that contains $N = |S|$ keys (in our case n-grams) that we wish to map to integers in the range 0 to $N - 1$, so that every key maps to a distinct integer (no collisions).

The first step is to use a hash function $g(x)$, to map every key to a bucket B in the range 0 to r. (For this step we use a simple hash function like the one used for generating fingerprints in the pervious section.)

$$B_i = \{x \in S | g(x) = i\} \; 0 \leq i \leq r$$

The function $g(x)$ is not perfect so several keys can map to the same bucket. Here we choose $r \leq N$, so that the number of buckets is less than or equal to the number of keys (to achieve 2.07 bits per key we use $r = \frac{N}{5}$, so that the average bucket size is 5). The buckets are then sorted into descending order according to the number of keys in each bucket $|B_i|$.

For the next step, a bit array, T, of size N is initialized to contain all zeros $T[0 \ldots N - 1]$. This bit array is used during construction to keep track of which integers in the range 0 to $N - 1$ the minimal perfect hash has already mapped keys to. Next we must assume we have access to a family of random and independent hash functions $h_1, h_2, h_3, \ldots$ that can be accessed using an integer index. In practice it sufficient to use functions that behave similarly to fully random independent hash functions and Belazzougui et al. (2009) demonstrate how such functions can be generated easily by combining two simple hash functions.

Next is the “displacement” step. For each bucket, in the sorted order from largest to smallest, they search for a random hash function that maps all elements of the bucket to values in T that are currently set to 0. Once this function has been found those positions in T are set to 1. So, for each bucket B_i, it is necessary to iteratively try hash functions, h_ℓ for $\ell = 1, 2, 3, \ldots$ to hash every element of B_i to a distinct index j in T that contains a zero.

$$\{h_\ell(x)|x \in B_i\} \cap \{j|T[j] = 1\} = \emptyset$$

where the size of $\{h_\ell(x)|x \in B_i\}$ is equal to the size of B_i. When such a hash function is found we need only to store the index, ℓ, of the successful function in an array σ and set $T[j] = 1$ for all positions j that h_ℓ hashed to. Notice that the reason the largest buckets are handled first is because they have the most elements to displace and this is easier when the array T contains more empty positions (zeros).

The final step in the algorithm is to compress the σ array (which has length equal to the number of buckets $|B|$), retaining $O(1)$ access. This compression is achieved using simple variable length encoding with an index array (Fredriksson and Nikitin, 2007).

3.2 Fingerprint and Rank Array

The hash function used in Stage 1 is perfect, so it is guaranteed to return unique integers for seen n-grams, but our hash function will also return integer values in the range 0 to $N - 1$ for n-grams that have not been seen before (were not used to build the hash function). To reduce the probability of these unseen n-grams giving false positives results from our model we store a fingerprint of each n-gram in Stage 2 of our structure that can be compared against the fingerprints of unseen n-grams when queried. If these fingerprints of the queried n-gram and the stored n-gram do not match then the model will correctly report that the n-gram has not been seen before. The size of this fingerprint determines the rate of false positives. Assuming that the fingerprint is generated by a random hash function, and that the returned integer of an *unseen* key from the MPH function is also random, expected false positive rate for the model is the same as the probability of two keys randomly hashing to the same value, $\frac{1}{2^m}$, where m is the number of bits of the fingerprint. The fingerprint can be generated using any suitably random hashing algorithm. We use Austin Appleby's Murmurhash2[1] implementation to fingerprint each n-gram and then store the m highest order bits. Stage 2 of the MPHR structure also stores a *rank* for every n-gram along with the fingerprint. This rank is an index into the array of Stage 3 of our structure that holds the unique values associated with any n-gram.

3.3 Unique Value Array

We describe our storage of the values associated with n-grams in our model assuming we are storing frequency "counts" of n-grams, but it applies also to storing quantized probabilities. For every n-gram, we store the 'rank' of the frequency count $r(key)$, ($r(key) \in [0...R - 1]$) and use a separate array in Stage 3 to store the frequency count value. This is similar to quantization in that it reduces the number of bits required for storage, but unlike quantization it does not require a loss of any information. This was motivated by the sparsity of n-gram frequency counts in corpora in the sense that if we take the lowest n-gram frequency count and the highest n-gram frequency count then most of the integers in that range do not occur as a frequency count of any n-grams in the corpus. For example in the Google Web1T data, there are 3.8 billion unique n-grams with frequency counts ranging from 40 to 95 Billion yet these n-grams only have 770 thousand distinct frequency counts (see Table 2). Because we only store the frequency rank, to keep the precise frequency information we need only $\lceil \log_2 K \rceil$ bits per n-gram, where K is the number of distinct frequency counts. To keep all information in the Google Web1T data we need only $\lceil \log_2 771058 \rceil = 20$ bits per n-gram. Rather than the bits needed to store the maximum frequency count associated with an n-gram, $\lceil \log_2 \text{maxcount} \rceil$, which for Google Web1T would be $\lceil \log_2 95119665584 \rceil = 37$ bits per n-gram.

	unique n-grams	maximum n-gram frequency count	unique counts
1gm	$1,585,620$	$71,363,822$	$16,896$
2gm	$55,809,822$	$9,319,466$	$20,237$
3gm	$250,928,598$	$829,366$	$12,425$
4gm	$493,134,812$	$231,973$	$6,838$
5gm	$646,071,143$	$86,943$	$4,201$
Total	$1,447,529,995$	$71,363,822$	$60,487$

Table 1: n-gram frequency counts from Gigaword corpus

[1] http://murmurhash.googlepages.com/

	unique n-grams	maximum n-gram frequency count	unique counts
1gm	$13,588,391$	$95,119,665,584$	$238,592$
2gm	$314,843,401$	$8,418,225,326$	$504,087$
3gm	$977,069,902$	$6,793,090,938$	$408,528$
4gm	$1,313,818,354$	$5,988,622,797$	$273,345$
5gm	$1,176,470,663$	$5,434,417,282$	$200,079$
Total	$3,795,790,711$	$95,119,665,584$	$771,058$

Table 2: n-gram frequency counts from Google Web1T corpus

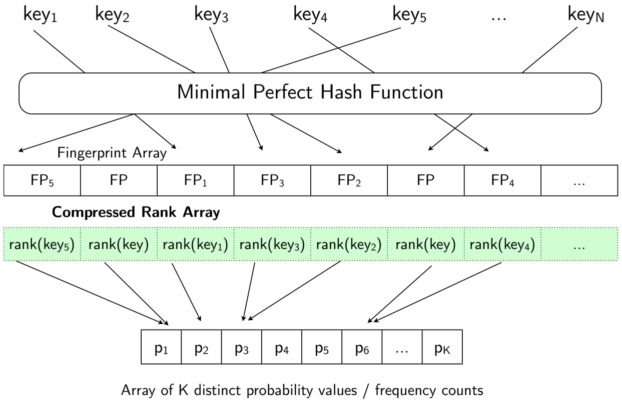


Figure 2: Compressed MPHR structure

3.4 Storage Requirements

We now consider the storage requirements of our S-MPHR approach, and how it compares against the Bloomier filter method of Talbot and Brants (2008). To start with, we put aside the gains that can come from using the ranking method, and instead consider just the costs of using the CHD approach for storing any language model. We saw that the storage requirements of the Talbot and Brants (2008) Bloomier filter method are a function of the number of n-grams n, the bits of data d to be stored per n-gram (with $d = v + e$: v bits for value storage, and e bits for error detection), and a multiplying factor of 1.23, giving an overall cost of $1.23d$ bits per n-gram. The cost for our basic approach is also easily computed. The explicit minimal PHF computed using the CHD algorithm brings a cost of 2.07 bits per n-gram for the PHF itself, and so the comparable overall cost to store a S-MPHR model is $2.07 + d$ bits per n-gram. For small values of d, the Bloomier filter approach has the smaller cost, but the 'break-even' point occurs when $d = 9$. When d is greater than 9 bits (as it usually will be), our approach wins out, being up to 18% more efficient.

The benefits that come from using the ranking method (Stage 3), for compactly storing count values, can only be evaluated in relation to the distributional characteristics specific corpora, for which we show results in Section 6.

4 Compressed MPHR Approach

Our second approach, called Compressed MPHR, further reduces the size of the model whilst maintaining O(1) time to query the model. Most compression techniques work by exploiting the redundancy in data. Our fingerprints are unfortunately random sequences of bits, so trying to compress these is fruitless, but the ranks associated with n-grams contain much redundancy and so are likely to compress well. We therefore modify our original architecture to put the ranks and fingerprints into separate arrays, of which the ranks array will be compressed, as shown in Figure 2.

Much like the final stage of the CHD minimal perfect hash algorithm we employ a random access compression algorithm of Fredriksson and Nikitin (2007) to reduce the size required by the array of ranks. This method allows compression while retaining O(1) access to query the model.

The first step in the compression is to encode the ranks array using a dense variable length coding. This coding works by assigning binary codes with different lengths to each number in the rank array, based on how frequent that number occurs. Let $s_1, s_2, s_3, \ldots, s_K$ be the ranks that occur in the rank array sorted by there frequency. Starting with most frequent number in the rank array (clearly 1 is the most common frequency count in the data unless it has been pruned) s_1 we assign it the bit code 0 and then assign s_2 the bit code 1, we then proceed by assigning bit codes of two bits, so s_3 is assigned 00, $s4$ is assigned 01, etc. until all two bit codes are used up. We then proceed to assign 3 bit codes and so on. All of the values from the rank array are coded in this form and concatenated to form a large bit vector retaining their original ordering. The length in bits for the ith number is thus $\lfloor \log_2 (i + 2) \rfloor$ and so the number of bits required for the whole variable length coded rank array is: $b = \sum_{i=0}^{K} f(s_i) \lfloor \log_2 (i + 2) \rfloor$. Where $f()$ gives the frequency that the rank occurs and K is the total number of distinct ranks. The code for the ith number is the binary representation with length $\lfloor \log_2{(i+2)} \rfloor$ of the number obtained using the formula:

$$\text{code} = i + 2 - 2^{\lfloor \log_2{(i+2)} \rfloor}$$

This variable length coded array is not useful by itself because we do not know where each number begins and ends, so we also store an index array hold this information. We create an additional bit array D of the same size b as the variable length coded array that simply contains ones in all positions that a code begins in the rank array and zeros in all other positions. That is the ith rank in the variable length coded array occurs at position $\text{select}_1(D, i)$, where select_1 gives the position of the ith one in the array. We do not actually store the D array, but instead we build a more space efficient structure to answer select_1 queries. Due the distribution of n-gram frequencies, the D array is typically dense in containing a large proportion of ones, so we build a *rank9sel* dictionary structure (Vigna, 2008) to answer these queries in constant time. We can use this structure to identify the ith code in our variable length encoded rank array by querying for its starting position, $\text{select}_1(D, i)$, and compute its length using its ending position, $\text{select}_1(D, i+1) - 1$. The code and its length can then be decoded to obtain the original rank:

$$\text{rank} = \text{code} + 2^{(\text{length in bits})} - 2$$

5 Tiered MPHR

In this section we describe an alternative route to extending our basic S-MPHR model to achieve better space efficiency, by using multiple hash stores. The method exploits distributional characteristics of the data, i.e. that lower rank values (those assigned to values shared by very many n-grams) are sufficient for representing the value information of a disproportionately large subset of the data. For the Google Web 1T data, for example, we find that the first 256 ranks account for nearly 85% of distinct n-grams, so if we could store ranks for these n-grams using only the 8 bits they require, whilst allowing perhaps 20 bits per n-gram for the remaining 15%, we would achieve an average of just under 10 bits per n-gram to store all the rank values.

To achieve this gain, we might *partition* the n-gram data into subsets requiring different amounts of space for value storage, and put these subsets in separate MPHRs, e.g. for the example just mentioned, with two MPHRs having 8 and 20 bit value storage respectively. Partitioning to a larger number h of MPHRs might further reduce this average cost. This simple approach has several problems. Firstly, it potentially requires a *series* of look up steps (i.e. up to h) to retrieve the value for any n-gram, with *all* hashes needing to be addressed to determine the unseen status of an unseen n-gram. Secondly, multiple look ups will produce a *compounding* of error rates, since we have up to h opportunities to falsely construe an unseen n-gram as seen, or to construe a seen n-gram as being stored in the wrong MPHR and so return an incorrect count for it.

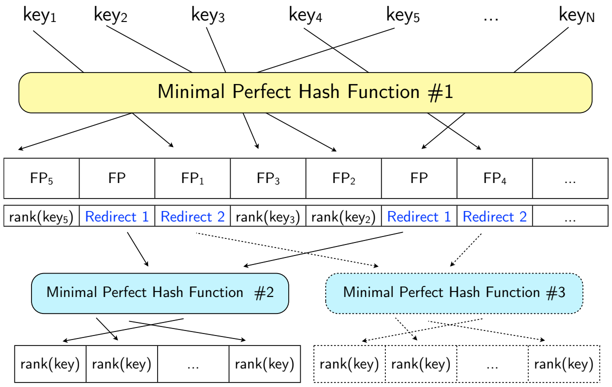


Figure 3: Tiered minimal perfect hash data structure

We will here explore an alternative approach that we call *Tiered* MPHR, which avoids this compounding of errors, and which limits the number of looks ups to a maximum of 2, irrespective of how many hashes are used. This approach employs a single *top-level* MPHR which has the full set of n-grams for its key-set, and stores a fingerprint for each. In addition, space is allocated to store rank values, but with some possible values being reserved to indicate *redirection* to other *secondary* hashes where values can be found. Each secondary hash has a minimal perfect hash function that is computed only for the n-grams whose values it stores. Secondary hashes do *not* need to record fingerprints, as fingerprint testing is done in the top-level hash.

For example, we might have a configuration of three hashes, with the top-level MPHR having 8-bit storage, and with secondary hashes having 10 and 20 bit storage respectively. Two values of the 8-bit store (e.g. 0 and 1) are reserved to indicate redirection to the specific secondary hashes, with the remaining values ($2\,..\,255$) representing ranks 1 to 254. The 10-bit secondary hash can store 1024 different values, which would then represent ranks 255 to 1278, with all ranks above this being represented in the 20-bit hash. To look up the count for an n-gram, we begin with the top-level hash, where fingerprint testing can immediately reject unseen n-grams. For some seen n-grams, the required rank value is provided directly by the top-level hash, but for others a redirection value is returned, indicating precisely the secondary hash in which the rank value will be found by simple look up (with no additional fingerprint testing). Figure 3 gives a generalized presentation of the structure of tiered MPHRs. Let us represent a configuration for a tiered MPHR as a sequence of bit values for their value stores, e.g. `(8,10,20)` for the example above, or $H = (b_1, \ldots .b_h)$ more generally (with b_1 being the top-level MPHR).

The overall memory cost of a particular configuration depends on distributional characteristics of the data stored. The top-level MPHR of configuration $(b_1, \ldots .b_h)$ stores all n-grams in its keyset, so its memory cost is calculated as before as $N \times (2.07 + m + b_1)$ (m the fingerprint size). The top-level MPHR must reserve $h - 1$ values for redirection, and so covers ranks $[1\,..\,(2^{b_1} - h + 1)]$. The second MPHR then covers the next 2^{b_2} ranks, starting at $(2^{b_1} - h + 2)$, and so on for further secondary MPHRs. This range of ranks determines the proportion μ_i of the overall n-gram set that each secondary MPHR b_i stores, and so the memory cost of each secondary MPHR is $N \times \mu_i \times (2.07 + b_i)$. The optimal T-MPHR configuration for a given data set is easily determined from distributional information (of the coverage of each rank), by a simple search.

6 Results

In this section, we present some results comparing the performance of our new storage methods to some of the existing methods, regarding the costs of storing LMs, and regarding the data access speeds that alternative systems allow.

Method	Gigaword		Web1T	
	full	quantized	full	quantized
Bloomier	6.00	3.08	7.53	3.08
S-MPHR	3.76	2.76	4.26	2.76
C-MPHR	2.19	2.09	3.40	2.09
T-MPHR	2.16	1.91	2.97	1.91

Table 3: Space usage in bytes/ngram using **12-bit** fingerprints and storing all 1 to 5 grams

Method	Gigaword		Web1T	
	full	quantized	full	quantized
Bloomier	5.38	2.46	6.91	2.46
S-MPHR	3.26	2.26	3.76	2.26
C-MPHR	1.69	1.59	2.90	1.59
T-MPHR	1.66	1.41	2.47	1.41

Table 4: Space usage in bytes/n-gram using **8-bit** fingerprints and storing all 1 to 5 grams

6.1 Comparison of memory costs

To test the effectiveness of our models we built models storing n-grams and full frequency counts for both the Gigaword and Google Web1T corpus storing all 1,2,3,4 and 5 grams. These corpora are very large, e.g. the Google Web1T corpus is 24.6GB when gzip compressed and contains over 3.7 billion n-grams, with frequency counts as large as 95 billion, requiring at least 37 bits to be stored. Using the Bloomier algorithm of Talbot and Brants (2008) with 37 bit values and 12 bit fingerprints would require 7.53 bytes/n-gram, so we would need 26.63GB to store a model for the entire corpus.

In comparison, our S-MPHR method requires only 4.26 bytes per n-gram to store full frequency count information and stores the entire Web1T corpus in just **15.05GB** or **57%** of the space required by the Bloomier method. This saving is mostly due to the ranking method allowing values to be stored at a cost of only 20 bits per n-gram. Applying the same rank array optimization to the Bloomier method significantly reduces its memory requirement, but S-MPHR still uses only 86% of the space that the Bloomier approach requires. Using T-MPHR instead, again with 12-bit fingerprints, we can store full counts for the Web 1T corpus in 10.50GB, which is small enough to be held in memory on many modern machines. Using 8-bit fingerprints, T-

Method	bytes/ ngram
SRILM Full, Compact	33.6
IRSTLM, 8-bit Quantized	9.1
Bloomier 12bit fp, 8bit Quantized	3.08
S-MPHR 12bit fp, 8bit Quantized	2.76
C-MPHR 12bit fp, 8bit Quantized	2.09
T-MPHR 12bit fp, 8bit Quantized	**1.91**

Table 5: Comparison between approaches for storing all 1 to 5 grams of the Gigaword Corpus

MPHR can store this data in just 8.74GB.

Tables 3, 4 and 5 show results for all methods[2] on both corpora, for storing full counts, and for when 8-bit binning quantization of counts is used.

6.2 Access speed comparisons

The three models we present in this paper perform queries in O(1) time and are thus asymptotically optimal, but this does not guarantee they perform well in practice, therefore in this section we measure query speed on a large set of n-grams and compare it to that of modern language modeling toolkits. We build a model of all unigrams and bigrams in the Gigaword corpus (see Table 1) using the C-MPHR method, SRILM (Stolcke, 2002), IRSTLM (Federico and Cettolo, 2007), and randLM[3] (Talbot and Osborne, 2007a) toolkits. RandLM is a modern language modeling toolkit that uses Bloom filter based structures to store large language models and has been integrated so that it can be used as the language model storage for the Moses statistical machine translation system (Koehn et al., 2007). We use randLM with the *BloomMap* (Talbot and Talbot, 2008) storage structure option with 8 bit quantized values and an error rate equivalent to using 8 bit fingerprints (as recommended in the Moses documentation). All methods are implemented in C++ and are run on a machine with 2.80GHz Intel Xeon E5462 processor and 64 GB of RAM. In addition we show a comparison to using a modern database, MySQL 5.0, to store the same data. We measure the speed of querying all models for the 55 million distinct bigrams that occur in the Gigaword, these results are shown in Table 6. Unsurprisingly all methods perform significantly faster than using a database as they build models that reside completely in memory. The C-MPHR method tested here is slower than both S-MPHR and T-MPHR models due to the extra operations required for access to the variable length encoded array yet still performs similarly to SRILM and IRSTLM and is 14.99 times faster than using randLM.

Test	Time (hr :min:sec)	Speed queries/sec
C-MPHR	00 : 01 : 50	507362
IRSTLM	00 : 02 : 12	422802
SRILM	00 : 01 : 29	627077
randLM	00 : 27 : 28	33865
MySQL 5	29 : 25 : 01	527

Table 6: Look-up speed performance comparison for C-MPHR and several other LM storage methods

7 Variable Length Fingerprints

To conclude our presentation of new methods for space-efficient language model storage, we suggest an additional possibility for reducing storage costs, which involves using different sizes of fingerprint for different n-grams. Recall that the only errors allowed by our approach are false-positives, i.e. where an unseen n-gram is falsely construed as being part of the model and a value returned for it. The idea behind using different sizes of fingerprint is that, intuitively, some possible errors seem worse than others, and in particular, it seems likely to be less damaging if we falsely construe an unseen n-gram as being a seen n-gram that has a low count or probability than as being one with a high count or probability.

False positives arise when our perfect hashing method maps an unseen n-gram to position where the stored n-gram fingerprint happens to coincide with that computed for the unseen n-gram. The risk of this occurring is a simple function of the size of fingerprints. To achieve a scheme that admits a higher risk of less damaging errors, but enforces a lower risk of more damaging errors, we need only store shorter fingerprints for n-grams in our model that have low counts or probabilities, and longer fingerprints for n-grams with higher values. This

[2] All T-MPHR results are for optimal configurations: Gigaword full: `(2,3,16)`, Gigaword quant: `(1,8)`, Web1T full: `(8,6,7,8,9,10,13,20)`, Web1T quant: `(1,8)`.

[3] `http://sourceforge.net/projects/randlm/`

idea could be implemented in different ways, e.g. by storing fingerprints of different lengths contiguously within a bit array, and constructing a 'selection structure' of the kind described in Section 4 to allow us to locate a given fingerprint within the bit array.

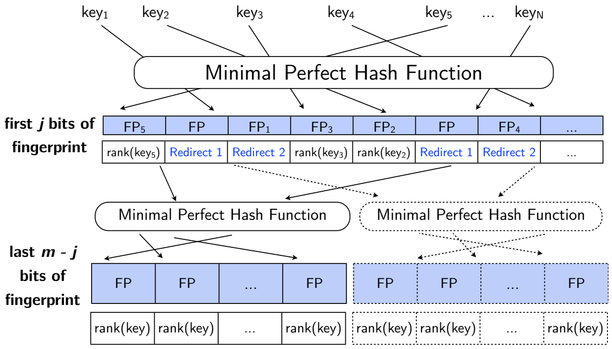


Figure 4: Variable length fingerprint T-MPHR structure using j bit fingerprints for the n-grams which are most rare and m bit fingerprints for all others.

We here instead consider an alternative implementation, based on the use of tiered structures. Recall that for T-MPHR, the top-level MPHR has all n-grams of the model as keys, and stores a fingerprint for each, plus a value that may represent an n-gram's count or probability, or that may redirect to a second-level hash where that information can be found. Redirection is done for items with higher counts or probabilities, so we can achieve lower error rates for precisely these items by storing *additional* fingerprint information for them in the second-level hash (see Figure 4). For example, we might have a top-level hash with only 4-bit fingerprints, but have an additional 8-bits of fingerprint for items also stored in a second-level hash, so there is quite a high risk (close to $\frac{1}{16}$) of returning a low count for an unseen n-gram, but a much lower risk of returning any higher count. Table 7 applies this idea to storing full and quantized counts of the Gigaword and Web 1T models, when fingerprints in the top-level MPHR have sizes in the range 1 to 6 bits, with the fingerprint information for items stored in secondary hashes being 'topped up' to 12 bits. This approach achieves storage costs of around 1 byte per n-gram or less for the quantized models.

Bits in lowest fingerprint	Gigaword Quantized	Web1T Quantized	Gigaword All	Web1T All
1	0.55	0.55	1.00	1.81
2	0.68	0.68	1.10	1.92
3	0.80	0.80	1.21	2.02
4	0.92	0.92	1.31	2.13
5	1.05	1.04	1.42	2.23
6	1.17	1.17	1.52	2.34

Table 7: Bytes per fingerprint for T-MPHR model using 1 to 6 bit fingerprints for rarest n-grams and 12 bit (in total) fingerprints for all other n-grams. (All configurations are as in Footnote 2.)

8 Conclusion

We have presented novel methods of storing large language models, consisting of billions of n-grams, that allow for quantized values or frequency counts to be accessed quickly and which require far less space than all known approaches. We show that it is possible to store all 1 to 5 grams in the Gigaword corpus, with full count information at a cost of just 1.66 bytes per n-gram, or with quantized counts for just 1.41 bytes per n-gram. We have shown that our models allow n-gram look-up at speeds comparable to modern language modeling toolkits (which have much greater storage costs), and at a rate approximately 15 times faster than a competitor approach for space-efficient storage.

References

Djamal Belazzougui, Fabiano Botelho, and Martin Dietzfelbinger. 2009. Hash, displace, and compress. *Algorithms - ESA 2009*, pages 682–693.

Burton H. Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426.

Thorsten Brants and Alex Franz. 2006. Google Web 1T 5-gram Corpus, version 1. Linguistic Data Consortium, Philadelphia, Catalog Number LDC2006T13, September.

Bernard Chazelle, Joe Kilian, Ronitt Rubinfeld, and Ayellet Tal. 2004. The bloomier filter: an efficient data structure for static support lookup tables. In *SODA '04*, pages 30–39, Philadelphia, PA, USA.

Philip Clarkson and Ronald Rosenfeld. 1997. Statistical language modeling using the CMU-cambridge toolkit. In *Proceedings of ESCA Eurospeech 1997*, pages 2707–2710.

Marcello Federico and Nicola Bertoldi. 2006. How many bits are needed to store probabilities for phrase-based translation? In *StatMT '06: Proceedings of the Workshop on Statistical Machine Translation*, pages 94–101, Morristown, NJ, USA. Association for Computational Linguistics.

Marcello Federico and Mauro Cettolo. 2007. Efficient handling of n-gram language models for statistical machine translation. In *StatMT '07: Proceedings of the Second Workshop on Statistical Machine Translation*, pages 88–95, Morristown, NJ, USA. Association for Computational Linguistics.

Edward Fredkin. 1960. Trie memory. *Commun. ACM*, 3(9):490–499.

Kimmo Fredriksson and Fedor Nikitin. 2007. Simple compression code supporting random access and fast string matching. In *Proc. of the 6th International Workshop on Efficient and Experimental Algorithms (WEA'07)*, pages 203–216.

Ulrich Germann, Eric Joanis, and Samuel Larkin. 2009. Tightly packed tries: How to fit large models into memory, and make them load fast, too. *Proceedings of the Workshop on Software Engineering, Testing, and Quality Assurance for Natural Language (SETQA-NLP 2009)*, pages 31–39.

Joshua Goodman and Jianfeng Gao. 2000. Language model size reduction by pruning and clustering. In *Proceedings of ICSLP'00*, pages 110–113.

David Graff. 2003. English Gigaword. Linguistic Data Consortium, catalog number LDC2003T05.

Boulos Harb, Ciprian Chelba, Jeffrey Dean, and Sanjay Ghemawat. 2009. Back-off language model compression. In *Proceedings of Interspeech*, pages 352–355.

Bo-June Hsu and James Glass. 2008. Iterative language model estimation:efficient data structure & algorithms. In *Proceedings of Interspeech*, pages 504–511.

F. Jelinek, B. Merialdo, S. Roukos, and M. Strauss I. 1990. Self-organized language modeling for speech recognition. In *Readings in Speech Recognition*, pages 450–506. Morgan Kaufmann.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *ACL '07: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180, Morristown, NJ, USA. Association for Computational Linguistics.

Andreas Stolcke. 1998. Entropy-based pruning of backoff language models. In *Proceedings of DARPA Broadcast News Transcription and Understanding Workshop*, pages 270–274.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, volume 2, pages 901–904, Denver.

David Talbot and Thorsten Brants. 2008. Randomized language models via perfect hash functions. *Proceedings of ACL-08 HLT*, pages 505–513.

David Talbot and Miles Osborne. 2007a. Randomised language modelling for statistical machine translation. In *Proceedings of ACL 07*, pages 512–519, Prague, Czech Republic, June.

David Talbot and Miles Osborne. 2007b. Smoothed bloom filter language models: Tera-scale LMs on the cheap. In *Proceedings of EMNLP*, pages 468–476.

David Talbot and John M. Talbot. 2008. Bloom maps. In *4th Workshop on Analytic Algorithmics and Combinatorics 2008 (ANALCO'08)*, pages 203—212, San Francisco, California.

David Talbot. 2009. Succinct approximate counting of skewed data. In *IJCAI'09: Proceedings of the 21st international jont conference on Artifical intelligence*, pages 1243–1248, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Sebastiano Vigna. 2008. Broadword implementation of rank/select queries. In *WEA'08: Proceedings of the 7th international conference on Experimental algorithms*, pages 154–168, Berlin, Heidelberg. Springer-Verlag.

Edward Whittaker and Bhinksha Raj. 2001. Quantization-based language model compression. Technical report, Mitsubishi Electric Research Laboratories, TR-2001-41.

Efficient Incremental Decoding for Tree-to-String Translation

Liang Huang [1]
[1]Information Sciences Institute
University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292, USA
{lhuang,haitaomi}@isi.edu

Haitao Mi [2,1]
[2]Key Lab. of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
P.O. Box 2704, Beijing 100190, China
htmi@ict.ac.cn

Abstract

Syntax-based translation models should *in principle* be efficient with polynomially-sized search space, but in practice they are often embarassingly slow, partly due to the cost of language model integration. In this paper we borrow from phrase-based decoding the idea to generate a translation *incrementally* left-to-right, and show that for tree-to-string models, with a clever encoding of derivation history, this method runs in average-case polynomial-time in theory, and linear-time with beam search in practice (whereas phrase-based decoding is exponential-time in theory and quadratic-time in practice). Experiments show that, with comparable translation quality, our tree-to-string system (in Python) can run more than 30 times faster than the phrase-based system Moses (in C++).

1 Introduction

Most efforts in statistical machine translation so far are variants of either phrase-based or syntax-based models. From a theoretical point of view, phrase-based models are neither expressive nor efficient: they typically allow arbitrary permutations and resort to language models to decide the best order. In theory, this process can be reduced to the Traveling Salesman Problem and thus requires an exponential-time algorithm (Knight, 1999). In practice, the decoder has to employ beam search to make it tractable (Koehn, 2004). However, even beam search runs in quadratic-time in general (see Sec. 2), unless a small *distortion limit* (say, d=5) further restricts the possible set of reorderings to those local ones by ruling out any long-distance reorderings that have a "jump" longer than d. This has been the standard practice with phrase-based models (Koehn et al., 2007), which fails to capture important long-distance reorderings like SVO-to-SOV.

	in theory	*in practice*
phrase-based	exponential	quadratic
tree-to-string	polynomial	linear

Table 1: [main result] Time complexity of our incremental tree-to-string decoding compared with phrase-based. In practice means "approximate search with beams."

Syntax-based models, on the other hand, use syntactic information to restrict reorderings to a computationally-tractable and linguistically-motivated subset, for example those generated by synchronous context-free grammars (Wu, 1997; Chiang, 2007). In theory the advantage seems quite obvious: we can now express global reorderings (like SVO-to-VSO) in polynomial-time (as opposed to exponential in phrase-based). But unfortunately, this polynomial complexity is super-linear (being generally cubic-time or worse), which is slow in practice. Furthermore, language model integration becomes more expensive here since the decoder now has to maintain target-language boundary words at *both* ends of a subtranslation (Huang and Chiang, 2007), whereas a phrase-based decoder only needs to do this at one end since the translation is always growing left-to-right. As a result, syntax-based models are often embarassingly slower than their phrase-based counterparts, preventing them from becoming widely useful.

Can we combine the merits of both approaches? While other authors have explored the possibilities

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 273–283,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

of enhancing phrase-based decoding with syntax-aware reordering (Galley and Manning, 2008), we are more interested in the other direction, i.e., can syntax-based models learn from phrase-based decoding, so that they still model global reordering, but in an efficient (preferably linear-time) fashion?

Watanabe et al. (2006) is an early attempt in this direction: they design a phrase-based-style decoder for the hierarchical phrase-based model (Chiang, 2007). However, this algorithm even with the beam search still runs in quadratic-time in practice. Furthermore, their approach requires grammar transformation that converts the original grammar into an equivalent binary-branching Greibach Normal Form, which is not always feasible in practice.

We take a fresh look on this problem and turn our focus to one particular syntax-based paradigm, tree-to-string translation (Liu et al., 2006; Huang et al., 2006), since this is the simplest and fastest among syntax-based approaches. We develop an incremental dynamic programming algorithm and make the following contributions:

- we show that, unlike previous work, our incremental decoding algorithm runs in average-case **polynomial-time** in theory for tree-to-string models, and the beam search version runs in **linear-time** in practice (see Table 1);
- large-scale experiments on a tree-to-string system confirm that, with comparable translation quality, our incremental decoder (in Python) can run more than 30 times faster than the phrase-based system Moses (in C++) (Koehn et al., 2007);
- furthermore, on the same tree-to-string system, incremental decoding is slightly faster than the standard cube pruning method at the same level of translation quality;
- this is also the first linear-time incremental decoder that performs global reordering.

We will first briefly review phrase-based decoding in this section, which inspires our incremental algorithm in the next section.

2 Background: Phrase-based Decoding

We will use the following running example from Chinese to English to explain both phrase-based and syntax-based decoding throughout this paper:

$_0$ *Bùshí* $_1$	*yǔ* $_2$ *Shālóng* $_3$	*jǔxíng* $_4$	*le* $_5$	*huìtán* $_6$
Bush	with Sharon	hold	-ed	meeting

'Bush held talks with Sharon'

2.1 Basic Dynamic Programming Algorithm

Phrase-based decoders generate partial target-language outputs in left-to-right order in the form of *hypotheses* (Koehn, 2004). Each hypothesis has a *coverage vector* capturing the source-language words translated so far, and can be extended into a longer hypothesis by a phrase-pair translating an uncovered segment. This process can be formalized as a deductive system. For example, the following deduction step grows a hypothesis by the phrase-pair ⟨*yǔ Shālóng*, with Sharon⟩ covering Chinese span [1-3]:

$$\frac{(\bullet\_\_\bullet\bullet\bullet_6) : (w, \text{“Bush held talks”})}{(\bullet\bullet\bullet_3\bullet\bullet\bullet) : (w', \text{“Bush held talks with Sharon”})} \quad (1)$$

where a $\bullet$ in the coverage vector indicates the source word at this position is "covered" and where w and $w' = w+c+d$ are the weights of the two hypotheses, respectively, with c being the cost of the phrase-pair, and d being the *distortion cost*. To compute d we also need to maintain the ending position of the last phrase (the $_3$ and $_6$ in the coverage vector).

To add a bigram model, we split each $-$LM item above into a series of $+$LM items; each $+$LM item has the form $(v,^{a}\ )$ where a is the last word of the hypothesis. Thus a $+$LM version of (1) might be:

$$\frac{(\bullet\_\_\bullet\bullet\bullet_6,^{\text{talks}}\ ) : (w, \text{“Bush held talks”})}{(\bullet\bullet\bullet_3\bullet\bullet\bullet,^{\text{Sharon}}\ ) : (w', \text{“Bush held talks with Sharon”})}$$

where the score of the resulting $+$LM item

$$w' = w + c + d - \log P_{lm}(\text{with} \mid \text{talk})$$

now includes a *combination cost* due to the bigrams formed when applying the phrase-pair. The complexity of this dynamic programming algorithm for g-gram decoding is $O(2^n n^2 |V|^{g-1})$ where n is the sentence length and $|V|$ is the English vocabulary size (Huang and Chiang, 2007).

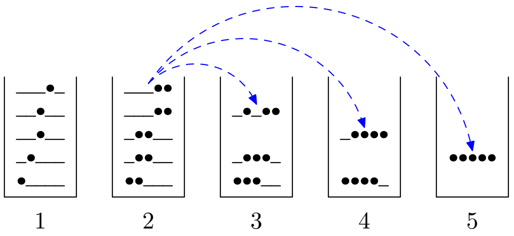


Figure 1: Beam search in phrase-based decoding expands the hypotheses in the current bin (#2) into longer ones.

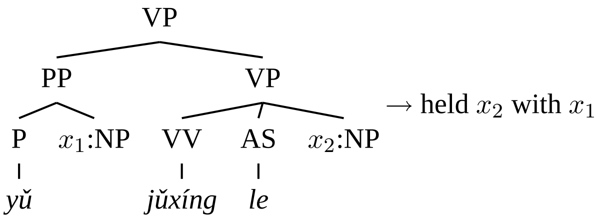


Figure 2: Tree-to-string rule r_3 for reordering.

2.2 Beam Search in Practice

To make the exponential algorithm practical, beam search is the standard approximate search method (Koehn, 2004). Here we group +LM items into n bins, with each bin B_i hosting at most b items that cover exactly i Chinese words (see Figure 1). The complexity becomes $O(n^2b)$ because there are a total of $O(nb)$ items in all bins, and to expand each item we need to scan the whole coverage vector, which costs $O(n)$. This quadratic complexity is still too slow in practice and we often set a small *distortion limit* of $d_{\max}$ (say, 5) so that no jumps longer than $d_{\max}$ are allowed. This method reduces the complexity to $O(nbd_{\max})$ but fails to capture long-distance reorderings (Galley and Manning, 2008).

3 Incremental Decoding for Tree-to-String Translation

We will first briefly review tree-to-string translation paradigm and then develop an incremental decoding algorithm for it inspired by phrase-based decoding.

3.1 Tree-to-string Translation

A typical tree-to-string system (Liu et al., 2006; Huang et al., 2006) performs translation in two 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.

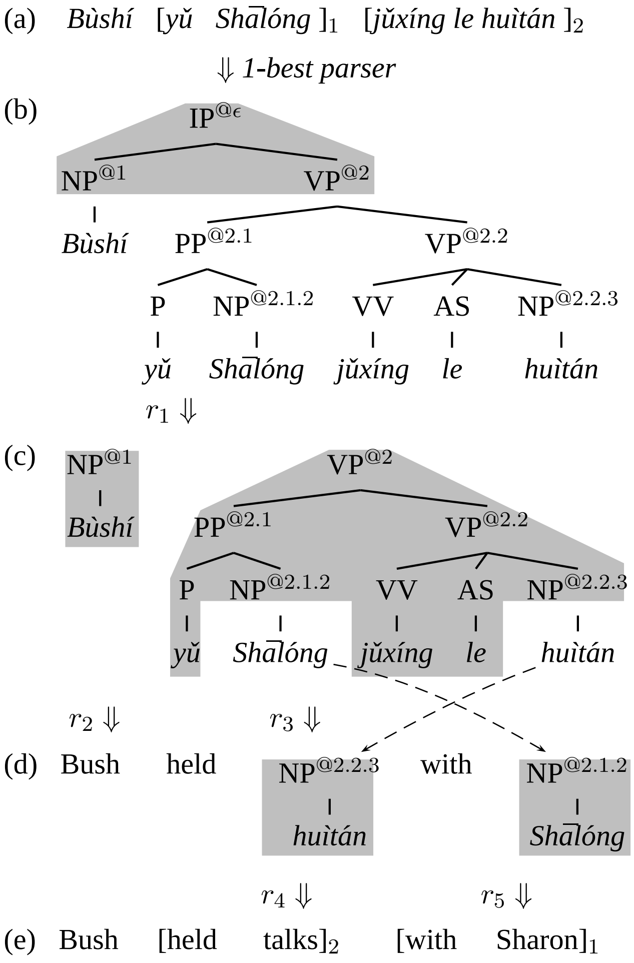


Figure 3: An example derivation of tree-to-string translation (much simplified from Mi et al. (2008)). Shaded regions denote parts of the tree that matches the rule.

Figure 3 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 the top-level word-order

(r_1) IP (x_1:NP x_2:VP) → x_1 x_2

which results in two unfinished subtrees, NP$^{@1}$ and VP$^{@2}$ in (c). Here $X^{@\eta}$ denotes a tree node of label X at tree address η (Shieber et al., 1995). (The root node has address ϵ, and the first child of node η has address η.1, etc.) Then rule r_2 grabs the *Bùshí* subtree and transliterate it into the English word

	in theory	in practice
phrase*	$O(2^n n^2 \cdot \|V\|^{g-1})$	$O(n^2 b)$
tree-to-str	$O(nc \cdot \|V\|^{4(g-1)})$	$O(ncb^2)$
*this work**	$O(n^{k \log^2(cr)} \cdot \|V\|^{g-1})$	$O(ncb)$

Table 2: Summary of time complexities of various algorithms. b is the beam width, V is the English vocabulary, and c is the number of translation rules per node. As a special case, phrase-based decoding with distortion limit $d_{\max}$ is $O(nbd_{\max})$. *: incremental decoding algorithms.

"Bush". Similarly, rule r_3 shown in Figure 2 is applied to the VP subtree, which swaps the two NPs, yielding the situation in (d). Finally two phrasal rules r_4 and r_5 translate the two remaining NPs and finish the translation.

In this framework, decoding without language model ($-$LM decoding) is simply a linear-time depth-first search with memoization (Huang et al., 2006), since a tree of n words is also of size $O(n)$ and we visit every node only once. Adding a language model, however, slows it down significantly because we now have to keep track of target-language boundary words, but unlike the phrase-based case in Section 2, here we have to remember both sides the leftmost and the rightmost boundary words: each node is now split into $+$LM items like $(\eta\ ^{a \star b})$ where η is a tree node, and a and b are left and right English boundary words. For example, a bigram $+$LM item for node VP$^{@2}$ might be

$$(\text{VP}^{@2\ \text{held} \star \text{Sharon}}).$$

This is also the case with other syntax-based models like Hiero or GHKM: language model integration overhead is the most significant factor that causes syntax-based decoding to be slow (Chiang, 2007). In theory $+$LM decoding is $O(nc|V|^{4(g-1)})$, where V denotes English vocabulary (Huang, 2007). In practice we have to resort to beam search again: at each node we would only allow top-b $+$LM items. With beam search, tree-to-string decoding with an integrated language model runs in time $O(ncb^2)$, where b is the size of the beam at each node, and c is (maximum) number of translation rules matched at each node (Huang, 2007). See Table 2 for a summary.

3.2 Incremental Decoding

Can we borrow the idea of phrase-based decoding, so that we also grow the hypothesis strictly left-to-right, and only need to maintain the rightmost boundary words?

The key intuition is to adapt the coverage-vector idea from phrase-based decoding to tree-to-string decoding. Basically, a coverage-vector keeps track of which Chinese spans have already been translated and which have not. Similarly, here we might need a "tree coverage-vector" that indicates which subtrees have already been translated and which have not. But unlike in phrase-based decoding, we can not simply choose any arbitrary uncovered subtree for the next step, since rules already dictate which subtree to visit next. In other words what we need here is not really a tree coverage vector, but more of a derivation history.

We develop this intuition into an agenda represented as a stack. Since tree-to-string decoding is a top-down depth-first search, we can simulate this recursion with a stack of active rules, i.e., rules that are not completed yet. For example we can simulate the derivation in Figure 3 as follows. At the root node IP$^{@\epsilon}$, we choose rule r_1, and push its English-side to the stack, with variables replaced by matched tree nodes, here x_1 for NP$^{@1}$ and x_2 for VP$^{@2}$. So we have the following stack

$$s = [\centerdot\ \text{NP}^{@1}\ \text{VP}^{@2}],$$

where the dot $\centerdot$ indicates the next symbol to process in the English word-order. Since node NP$^{@1}$ is the first in the English word-order, we expand it first, and push rule r_2 rooted at NP to the stack:

$$[\centerdot\ \text{NP}^{@1}\ \text{VP}^{@2}\ ]\ [\centerdot\ \text{Bush}].$$

Since the symbol right after the dot in the top rule is a word, we immediately grab it, and append it to the current hypothesis, which results in the new stack

$$[\centerdot\ \text{NP}^{@1}\ \text{VP}^{@2}\ ]\ [\text{Bush}\ \centerdot\ ].$$

Now the top rule on the stack has finished (dot is at the end), so we trigger a "pop" operation which pops the top rule and advances the dot in the second-to-top rule, denoting that NP$^{@1}$ is now completed:

$$[\text{NP}^{@1}\ \centerdot\ \text{VP}^{@2}].$$

	stack	hypothesis
	[<s> $\bullet$ IP$^{@\epsilon}$ </s>]	<s>
p	[<s> $\bullet$ IP$^{@\epsilon}$ </s>] [$\bullet$ NP$^{@1}$ VP$^{@2}$]	<s>
p	[<s> $\bullet$ IP$^{@\epsilon}$ </s>] [$\bullet$ NP$^{@1}$ VP$^{@2}$] [$\bullet$ Bush]	<s>
s	[<s> $\bullet$ IP$^{@\epsilon}$ </s>] [$\bullet$ NP$^{@1}$ VP$^{@2}$] [Bush $\bullet$]	<s> Bush
c	[<s> $\bullet$ IP$^{@\epsilon}$ </s>] [NP$^{@1}$ $\bullet$ VP$^{@2}$]	<s> Bush
p	[<s> $\bullet$ IP$^{@\epsilon}$ </s>] [NP$^{@1}$ $\bullet$ VP$^{@2}$] [$\bullet$ held NP$^{@2.2.3}$ with NP$^{@2.1.2}$]	<s> Bush
s	[<s> $\bullet$ IP$^{@\epsilon}$ </s>] [NP$^{@1}$ $\bullet$ VP$^{@2}$] [held $\bullet$ NP$^{@2.2.3}$ with NP$^{@2.1.2}$]	<s> Bush held
p	[<s> $\bullet$ IP$^{@\epsilon}$ </s>] [NP$^{@1}$ $\bullet$ VP$^{@2}$] [held $\bullet$ NP$^{@2.2.3}$ with NP$^{@2.1.2}$] [$\bullet$ talks]	<s> Bush held
s	[<s> $\bullet$ IP$^{@\epsilon}$ </s>] [NP$^{@1}$ $\bullet$ VP$^{@2}$] [held $\bullet$ NP$^{@2.2.3}$ with NP$^{@2.1.2}$] [talks $\bullet$]	<s> Bush held talks
c	[<s> $\bullet$ IP$^{@\epsilon}$ </s>] [NP$^{@1}$ $\bullet$ VP$^{@2}$] [held NP$^{@2.2.3}$ $\bullet$ with NP$^{@2.1.2}$]	<s> Bush held talks
s	[<s> $\bullet$ IP$^{@\epsilon}$ </s>] [NP$^{@1}$ $\bullet$ VP$^{@2}$] [held NP$^{@2.2.3}$ with $\bullet$ NP$^{@2.1.2}$]	<s> Bush held talks with
p	[<s> $\bullet$ IP$^{@\epsilon}$ </s>] [NP$^{@1}$ $\bullet$ VP$^{@2}$] [held NP$^{@2.2.3}$ with $\bullet$ NP$^{@2.1.2}$] [$\bullet$ Sharon]	<s> Bush held talks with
s	[<s> $\bullet$ IP$^{@\epsilon}$ </s>] [NP$^{@1}$ $\bullet$ VP$^{@2}$] [held NP$^{@2.2.3}$ with $\bullet$ NP$^{@2.1.2}$] [Sharon$\bullet$]	<s> Bush held talks with Sharon
c	[<s> $\bullet$ IP$^{@\epsilon}$ </s>] [NP$^{@1}$ $\bullet$ VP$^{@2}$] [held NP$^{@2.2.3}$ with NP$^{@2.1.2}$$\bullet$]	<s> Bush held talks with Sharon
c	[<s> $\bullet$ IP$^{@\epsilon}$ </s>] [NP$^{@1}$ VP$^{@2}$$\bullet$]	<s> Bush held talks with Sharon
c	[<s> IP$^{@\epsilon}$ $\bullet$ </s>]	<s> Bush held talks with Sharon
s	[<s> IP$^{@\epsilon}$ </s>$\bullet$]	<s> Bush held talks with Sharon </s>

Figure 4: Simulation of tree-to-string derivation in Figure 3 in the incremental decoding algorithm. Actions: p, predict; s, scan; c, complete (see Figure 5).

Item $\quad \ell : \langle s,\ \rho\rangle :\ w; \quad \ell$: step, s: stack, ρ: hypothesis, w: weight

Equivalence $\quad \ell : \langle s,\ \rho\rangle \sim \ell : \langle s',\ \rho'\rangle$ iff. $s = s'$ and $last_{g-1}(\rho) = last_{g-1}(\rho')$

Axiom
$$0 : \langle [\texttt{<s>}^{g-1} \bullet \epsilon\ \texttt{</s>}],\ \texttt{<s>}^{g-1}\rangle :\ 0$$

Predict
$$\frac{\ell : \langle ...\ [\alpha \bullet \eta\ \beta],\ \rho\rangle :\ w}{\ell + |C(r)| : \langle ...\ [\alpha \bullet \eta\ \beta]\ [\bullet\ f(\eta, E(r))],\ \rho\rangle :\ w + c(r)} \qquad match(\eta, C(r))$$

Scan
$$\frac{\ell : \langle ...\ [\alpha \bullet e\ \beta],\ \rho\rangle :\ w}{\ell : \langle ...\ [\alpha\ e \bullet \beta],\ \rho e\rangle :\ w - \log \Pr(e \mid last_{g-1}(\rho))}$$

Complete
$$\frac{\ell : \langle ...\ [\alpha \bullet \eta\ \beta]\ [\gamma \bullet],\ \rho\rangle :\ w}{\ell : \langle ...\ [\alpha\ \eta \bullet \beta],\ \rho\rangle :\ w}$$

Goal
$$|T| : \langle [\texttt{<s>}^{g-1}\ \epsilon\ \texttt{</s>}\bullet],\ \rho\texttt{</s>}\rangle :\ w$$

Figure 5: Deductive system for the incremental tree-to-string decoding algorithm. Function $last_{g-1}(\cdot)$ returns the rightmost $g - 1$ words (for g-gram LM), and $match(\eta, C(r))$ tests matching of rule r against the subtree rooted at node η. $C(r)$ and $E(r)$ are the Chinese and English sides of rule r, and function $f(\eta, E(r)) = [x_i \mapsto \eta.var(i)]E(r)$ replaces each variable x_i on the English side of the rule with the descendant node $\eta.var(i)$ under η that matches x_i.

The next step is to expand VP$^{@2}$, and we use rule r_3 and push its English-side "VP → held x_2 with x_1" onto the stack, again with variables replaced by matched nodes:

$$[\text{NP}^{@1} \bullet \text{VP}^{@2}]\ [\bullet\ \text{held NP}^{@2.2.3} \text{ with NP}^{@2.1.2}]$$

Note that this is a reordering rule, and the stack always follows the English word order because we generate hypothesis incrementally left-to-right. Figure 4 works out the full example.

We formalize this algorithm in Figure 5. Each item $\langle s,\ \rho\rangle$ consists of a stack s and a hypothesis ρ. Similar to phrase-based dynamic programming, only the last $g-1$ words of ρ are part of the signature for decoding with g-gram LM. Each stack is a list of *dotted rules*, i.e., rules with dot positions indicting progress, in the style of Earley (1970). We call the last (rightmost) rule on the stack the *top rule*, which is the rule being processed currently. The symbol after the dot in the top rule is called the *next symbol*, since it is the symbol to expand or process next. Depending on the next symbol a, we can perform one of the three actions:

- if a is a node η, we perform a Predict action which expands η using a rule r that can pattern-match the subtree rooted at η; we push r is to the stack, with the dot at the beginning;
- if a is an English word, we perform a Scan action which immediately adds it to the current hypothesis, advancing the dot by one position;
- if the dot is at the end of the top rule, we perform a Complete action which simply pops stack and advance the dot in the new top rule.

3.3 Polynomial Time Complexity

Unlike phrase-based models, we show here that incremental decoding runs in average-case polynomial-time for tree-to-string systems.

Lemma 1. *For an input sentence of n words and its parse tree of depth d, the worst-case complexity of our algorithm is $f(n,d) = c(cr)^d|V|^{g-1} = O((cr)^d n^{g-1})$, assuming relevant English vocabulary $|V| = O(n)$, and where constants c, r and g are the maximum number of rules matching each tree node, the maximum arity of a rule, and the language-model order, respectively.*

Proof. The time complexity depends (in part) on the number of all possible stacks for a tree of depth d. A stack is a list of rules covering a path from the root node to one of the leaf nodes in the following form:

$$\overbrace{[\dots \bullet\eta_1 \dots]}^{R_1}\ \overbrace{[\dots \bullet\eta_2 \dots]}^{R_2}\ \dots\ \overbrace{[\dots \bullet\eta_s \dots]}^{R_s},$$

where $\eta_1 = \epsilon$ is the root node and η_s is a leaf node, with stack depth $s \le d$. Each rule $R_i (i > 1)$ expands node η_{i-1}, and thus has c choices by the definition of grammar constant c. Furthermore, each rule in the stack is actually a dotted-rule, i.e., it is associated with a dot position ranging from 0 to r, where r is the arity of the rule (length of English side of the rule). So the total number of stacks is $O((cr)^d)$.

Besides the stack, each state also maintains $(g-1)$ rightmost words of the hypothesis as the language model signature, which amounts to $O(|V|^{g-1})$. So the total number of states is $O((cr)^d|V|^{g-1})$. Following previous work (Chiang, 2007), we assume a constant number of English translations for each foreign word in the input sentence, so $|V| = O(n)$. And as mentioned above, for each state, there are c possible expansions, so the overall time complexity is $f(n,d) = c(cr)^d|V|^{g-1} = O((cr)^d n^{g-1})$. □

We do average-case analysis below because the tree depth (height) for a sentence of n words is a random variable: in the worst-case it can be linear in n (degenerated into a linear-chain), but we assume this adversarial situation does not happen frequently, and the average tree depth is $O(\log n)$.

Theorem 1. *Assume for each n, the depth of a parse tree of n words, notated d_n, distributes normally with logarithmic mean and variance, i.e., $d_n \sim \mathcal{N}(\mu_n, \sigma_n^2)$, where $\mu_n = O(\log n)$ and $\sigma_n^2 = O(\log n)$, then the average-case complexity of the algorithm is $h(n) = O(n^{k\log^2(cr)+g-1})$ for constant k, thus polynomial in n.*

Proof. From Lemma 1 and the definition of average-case complexity, we have

$$h(n) = \mathbb{E}_{d_n \sim \mathcal{N}(\mu_n, \sigma_n^2)}[f(n, d_n)],$$

where $\mathbb{E}_{x\sim D}[\cdot]$ denotes the expectation with respect to the random variable x in distribution D.

$$\begin{aligned} h(n) &= \mathbb{E}_{d_n \sim \mathcal{N}(\mu_n, \sigma_n^2)}[f(n, d_n)] \\ &= \mathbb{E}_{d_n \sim \mathcal{N}(\mu_n, \sigma_n^2)}[O((cr)^{d_n} n^{g-1})], \\ &= O(n^{g-1} \mathbb{E}_{d_n \sim \mathcal{N}(\mu_n, \sigma_n^2)}[(cr)^{d_n}]), \\ &= O(n^{g-1} \mathbb{E}_{d_n \sim \mathcal{N}(\mu_n, \sigma_n^2)}[\exp(d_n \log(cr))]) \quad (2) \end{aligned}$$

Since $d_n \sim \mathcal{N}(\mu_n, \sigma_n^2)$ is a normal distribution, $d_n \log(cr) \sim \mathcal{N}(\mu', \sigma'^2)$ is also a normal distribution, where $\mu' = \mu_n \log(cr)$ and $\sigma' = \sigma_n \log(cr)$. Therefore $\exp(d_n \log(cr))$ is a log-normal distribution, and by the property of log-normal distribution, its expectation is $\exp(\mu' + \sigma'^2/2)$. So we have

$$\begin{aligned} & \mathbb{E}_{d_n \sim \mathcal{N}(\mu_n, \sigma^2/2)}[\exp(d_n \log(cr))] \\ = \; & \exp(\mu' + \sigma'^2/2) \\ = \; & \exp(\mu_n \log(cr) + \sigma_n^2 \log^2(cr)/2) \\ = \; & \exp(O(\log n) \log(cr) + O(\log n) \log^2(cr)/2) \\ = \; & \exp(O(\log n) \log^2(cr)) \\ \leq \; & \exp(k(\log n) \log^2(cr)), \qquad \text{for some constant } k \\ = \; & \exp(\log n^{k \log^2(cr)}) \\ = \; & n^{k \log^2(cr)}. \qquad (3) \end{aligned}$$

Plug it back to Equation (2), and we have the average-case complexity

$$\begin{aligned} \mathbb{E}_{d_n}[f(n, d_n)] &\leq O(n^{g-1} n^{k \log^2(cr)}) \\ &= O(n^{k \log^2(cr) + g - 1}). \qquad (4) \end{aligned}$$

Since k, c, r and g are constants, the average-case complexity is polynomial in sentence length n. □

The assumption $d_n \sim \mathcal{N}(O(\log n), O(\log n))$ will be empirically verified in Section 5.

3.4 Linear-time Beam Search

Though polynomial complexity is a desirable property in theory, the degree of the polynomial, $O(\log cr)$ might still be too high in practice, depending on the translation grammar. To make it linear-time, we apply the beam search idea from phrase-based again. And once again, the only question to decide is the choice of "binning": how to assign each item to a particular bin, depending on their progress?

While the number of Chinese words covered is a natural progress indicator for phrase-based, it does not work for tree-to-string because, among the three actions, only scanning grows the hypothesis. The prediction and completion actions do not make real progress in terms of *words*, though they do make progress on the *tree*. So we devise a novel progress indicator natural for tree-to-string translation: the number of tree nodes covered so far. Initially that number is zero, and in a prediction step which expands node η using rule r, the number increments by $|C(r)|$, the size of the Chinese-side treelet of r. For example, a prediction step using rule r_3 in Figure 2 to expand VP$^{@2}$ will increase the tree-node count by $|C(r_3)| = 6$, since there are six tree nodes in that rule (not counting leaf nodes or variables).

Scanning and completion do not make progress in this definition since there is no new tree node covered. In fact, since both of them are deterministic operations, they are treated as "closure" operators in the real implementation, which means that after a prediction, we always do as many scanning/completion steps as possible until the symbol after the dot is another node, where we have to wait for the next prediction step.

This method has $|T| = O(n)$ bins where $|T|$ is the size of the parse tree, and each bin holds b items. Each item can expand to c new items, so the overall complexity of this beam search is $O(ncb)$, which is linear in sentence length.

4 Related Work

The work of Watanabe et al. (2006) is closest in spirit to ours: they also design an incremental decoding algorithm, but for the hierarchical phrase-based system (Chiang, 2007) instead. While we leave detailed comparison and theoretical analysis to a future work, here we point out some obvious differences:

1. due to the difference in the underlying translation models, their algorithm runs in $O(n^2 b)$ time with beam search in practice while ours is linear. This is because each prediction step now has $O(n)$ choices, since they need to expand nodes like VP[1, 6] as:

   $$\text{VP[1,6]} \rightarrow \text{PP[1, } i \text{]} \;\; \text{VP[} i \text{, 6]},$$

   where the midpoint i in general has $O(n)$ choices (just like in CKY). In other words, their grammar constant c becomes $O(n)$.

2. different binning criteria: we use the number of tree nodes covered, while they stick to the orig-

inal phrase-based idea of number of Chinese words translated;

3. as a result, their framework requires grammar transformation into the binary-branching Greibach Normal Form (which is not always possible) so that the resulting grammar always contain at least one Chinese word in each rule in order for a prediction step to always make progress. Our framework, by contrast, works with any grammar.

Besides, there are some other efforts less closely related to ours. As mentioned in Section 1, while we focus on enhancing syntax-based decoding with phrase-based ideas, other authors have explored the reverse, but also interesting, direction of enhancing phrase-based decoding with syntax-aware reordering. For example Galley and Manning (2008) propose a shift-reduce style method to allow hieararchical non-local reorderings in a phrase-based decoder. While this approach is certainly better than pure phrase-based reordering, it remains quadratic in run-time with beam search.

Within syntax-based paradigms, cube pruning (Chiang, 2007; Huang and Chiang, 2007) has become the standard method to speed up $+$LM decoding, which has been shown by many authors to be highly effective; we will be comparing our incremental decoder with a baseline decoder using cube pruning in Section 5. It is also important to note that cube pruning and incremental decoding are not mutually exclusive, rather, they could potentially be combined to further speed up decoding. We leave this point to future work.

Multipass coarse-to-fine decoding is another popular idea (Venugopal et al., 2007; Zhang and Gildea, 2008; Dyer and Resnik, 2010). In particular, Dyer and Resnik (2010) uses a two-pass approach, where their first-pass, $-$LM decoding is also incremental and polynomial-time (in the style of Earley (1970) algorithm), but their second-pass, $+$LM decoding is still bottom-up CKY with cube pruning.

5 Experiments

To test the merits of our incremental decoder we conduct large-scale experiments on a state-of-the-art tree-to-string system, and compare it with the standard phrase-based system of Moses. Furturemore we also compare our incremental decoder with the standard cube pruning approach on the same tree-to-string decoder.

5.1 Data and System Preparation

Our training corpus consists of 1.5M sentence pairs with about 38M/32M words in Chinese/English, respectively. We first word-align them by GIZA++ and then parse the Chinese sentences using the Berkeley parser (Petrov and Klein, 2007), then apply the GHKM algorithm (Galley et al., 2004) to extract tree-to-string translation rules. We use SRILM Toolkit (Stolcke, 2002) to train a trigram language model with modified Kneser-Ney smoothing on the target side of training corpus. At decoding time, we again parse the input sentences into trees, and convert them into translation forest by rule pattern-matching (Mi et al., 2008).

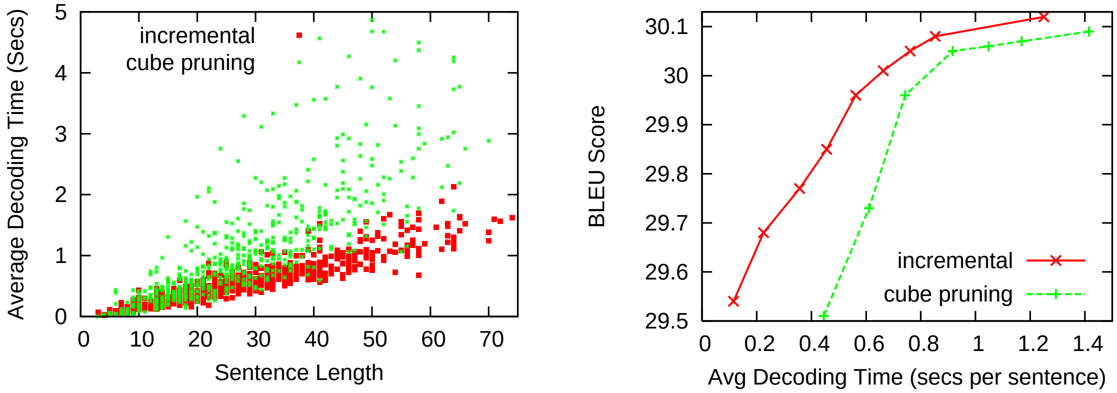
We use the newswire portion of 2006 NIST MT Evaluation test set (616 sentences) as our development set and the newswire portion of 2008 NIST MT Evaluation test set (691 sentences) as our test set. We evaluate the translation quality using the BLEU-4 metric, which is calculated by the script mteval-v13a.pl with its default setting which is case-insensitive matching of n-grams. We use the standard minimum error-rate training (Och, 2003) to tune the feature weights to maximize the system's BLEU score on development set.

We first verify the assumptions we made in Section 3.3 in order to prove the theorem that tree depth (as a random variable) is normally-distributed with $O(\log n)$ mean and variance. Qualitatively, we verified that for most n, tree depth $d(n)$ does look like a normal distribution. Quantitatively, Figure 6 shows that average tree height correlates extremely well with $3.5 \log n$, while tree height variance is bounded by $5.5 \log n$.

5.2 Comparison with Cube pruning

We implemented our incremental decoding algorithm in Python, and test its performance on the development set. We first compare it with the standard cube pruning approach (also implemented in Python) on the same tree-to-string system.[1] Fig-

[1] Our implementation of cube pruning follows (Chiang, 2007; Huang and Chiang, 2007) where besides a beam size b of unique $+$LM items, there is also a hard limit (of 1000) on the



(a) decoding time against sentence length

(b) BLEU score against decoding time

Figure 7: Comparison with cube pruning. The scatter plot in (a) confirms that our incremental decoding scales linearly with sentence length, while cube pruning super-linearly ($b = 50$ for both). The comparison in (b) shows that at the same level of translation quality, incremental decoding is slightly faster than cube pruning, especially at smaller beams.

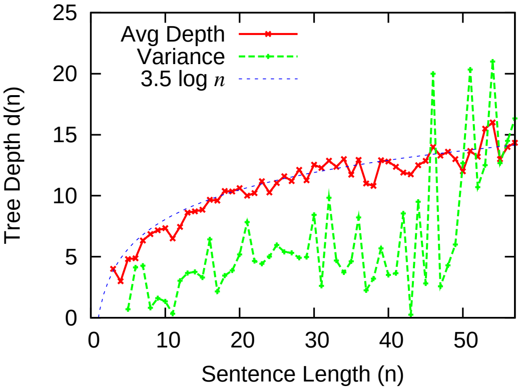


Figure 6: Mean and variance of tree depth vs. sentence length. The mean depth clearly scales with $3.5 \log n$, and the variance is bounded by $5.5 \log n$.

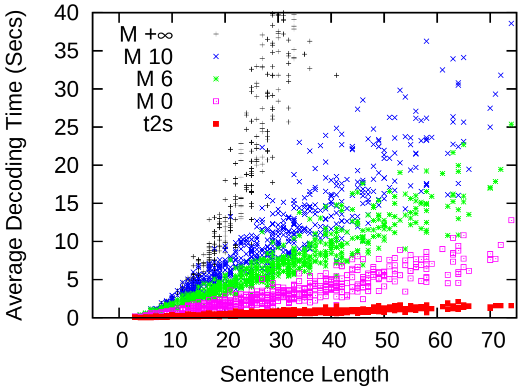


Figure 8: Comparison of our incremental tree-to-string decoder with Moses in terms of speed. Moses is shown with various distortion limits (0, 6, 10, $+\infty$; optimal: 10).

ure 7(a) is a scatter plot of decoding times versus sentence length (using beam $b = 50$ for both systems), where we confirm that our incremental decoder scales linearly, while cube pruning has a slight tendency of superlinearity. Figure 7(b) is a side-by-side comparison of decoding speed versus translation quality (in BLEU scores), using various beam sizes for both systems (b=10–70 for cube pruning, and b=10–110 for incremental). We can see that incremental decoding is slightly faster than cube pruning at the same levels of translation quality, and the difference is more pronounced at smaller beams: for example, at the lowest levels of translation quality (BLEU scores around 29.5), incremental decoding takes only 0.12 seconds, which is about 4 times as fast as cube pruning. We stress again that cube pruning and incremental decoding are not mutually exclusive, and rather they could potentially be combined to further speed up decoding.

5.3 Comparison with Moses

We also compare with the standard phrase-based system of Moses (Koehn et al., 2007), with standard settings except for the ttable limit, which we set to 100. Figure 8 compares our incremental decoder

number of (non-unique) pops from priority queues.

system/decoder	BLEU	time
Moses (optimal $d_{\max}$=10)	29.41	10.8
tree-to-str: cube pruning (b=10)	29.51	0.65
tree-to-str: cube pruning (b=20)	29.96	0.96
tree-to-str: incremental (b=10)	29.54	0.32
tree-to-str: incremental (b=50)	29.96	0.77

Table 3: Final BLEU score and speed results on the test data (691 sentences), compared with Moses and cube pruning. Time is in seconds per sentence, including parsing time (0.21s) for the two tree-to-string decoders.

with Moses at various distortion limits ($d_{\max}$=0, 6, 10, and $+\infty$). Consistent with the theoretical analysis in Section 2, Moses with no distortion limit ($d_{\max} = +\infty$) scale *quadratically*, and monotone decoding ($d_{\max} = 0$) scale linearly. We use MERT to tune the best weights for each distortion limit, and $d_{\max} = 10$ performs the best on our dev set.

Table 3 reports the final results in terms of BLEU score and speed on the test set. Our linear-time incremental decoder with the small beam of size $b = 10$ achieves a BLEU score of 29.54, comparable to Moses with the optimal distortion limit of 10 (BLEU score 29.41). But our decoding (including source-language parsing) only takes 0.32 seconds a sentences, which is more than 30 times faster than Moses. With a larger beam of $b = 50$ our BLEU score increases to 29.96, which is a half BLEU point better than Moses, but still about 15 times faster.

6 Conclusion

We have presented an incremental dynamic programming algorithm for tree-to-string translation which resembles phrase-based based decoding. This algorithm is the first incremental algorithm that runs in polynomial-time in theory, and linear-time in practice with beam search. Large-scale experiments on a state-of-the-art tree-to-string decoder confirmed that, with a comparable (or better) translation quality, it can run more than 30 times faster than the phrase-based system of Moses, even though ours is in Python while Moses in C++. We also showed that it is slightly faster (and scale better) than the popular cube pruning technique. For future work we would like to apply this algorithm to forest-based translation and hierarchical system by pruning the first-pass $-$LM forest. We would also combine cube pruning with our incremental algorithm, and study its performance with higher-order language models.

Acknowledgements

We would like to thank David Chiang, Kevin Knight, and Jonanthan Graehl for discussions and the anonymous reviewers for comments. In particular, we are indebted to the reviewer who pointed out a crucial mistake in Theorem 1 and its proof in the submission. This research was supported in part by DARPA, under contract HR0011-06-C-0022 under subcontract to BBN Technologies, and under DOI-NBC Grant N10AP20031, and in part by the National Natural Science Foundation of China, Contracts 60736014 and 90920004.

References

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–208.

Chris Dyer and Philip Resnik. 2010. Context-free reordering, finite-state translation. In *Proceedings of NAACL*.

Jay Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102.

Michel Galley and Christopher D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *Proceedings of EMNLP 2008*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT-NAACL*, pages 273–280.

Liang Huang and David Chiang. 2007. Forest rescoring: Fast decoding with integrated language models. In *Proceedings of ACL*, Prague, Czech Rep., 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. 2007. Binarization, synchronous binarization, and target-side binarization. In *Proc. NAACL Workshop on Syntax and Structure in Statistical Translation*.

Kevin Knight. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615.

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 *Proceedings of ACL: demonstration sesion*.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proceedings of AMTA*, pages 115–124.

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.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of ACL: HLT*, Columbus, OH.

Franz Joseph Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of HLT-NAACL*.

Stuart Shieber, Yves Schabes, and Fernando Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24:3–36.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *Proceedings of ICSLP*, volume 30, pages 901–904.

Ashish Venugopal, Andreas Zollmann, and Stephen Vogel. 2007. An efficient two-pass approach to synchronous-CFG driven statistical MT. In *Proceedings of HLT-NAACL*.

Taro Watanabe, Hajime Tsukuda, and Hideki Isozaki. 2006. Left-to-right target generation for hierarchical phrase-based translation. In *Proceedings of COLING-ACL*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404.

Hao Zhang and Daniel Gildea. 2008. Efficient multi-pass decoding for synchronous context free grammars. In *Proceedings of ACL*.

Modeling Perspective using Adaptor Grammars

Eric A. Hardisty
Department of Computer Science
and UMIACS
University of Maryland
College Park, MD
hardisty@cs.umd.edu

Jordan Boyd-Graber
UMD iSchool
and UMIACS
University of Maryland
College Park, MD
jbg@umiacs.umd.edu

Philip Resnik
Department of Linguistics
and UMIACS
University of Maryland
College Park, MD
resnik@umd.edu

Abstract

Strong indications of perspective can often come from collocations of arbitrary length; for example, someone writing *get the government out of my X* is typically expressing a conservative rather than progressive viewpoint. However, going beyond unigram or bigram features in perspective classification gives rise to problems of data sparsity. We address this problem using nonparametric Bayesian modeling, specifically adaptor grammars (Johnson et al., 2006). We demonstrate that an *adaptive naïve Bayes* model captures multiword lexical usages associated with perspective, and establishes a new state-of-the-art for perspective classification results using the Bitter Lemons corpus, a collection of essays about mid-east issues from Israeli and Palestinian points of view.

1 Introduction

Most work on the computational analysis of sentiment and perspective relies on lexical features. This makes sense, since an author's choice of words is often used to express overt opinions (e.g. describing healthcare reform as *idiotic* or *wonderful*) or to frame a discussion in order to convey a perspective more implicitly (e.g. using the term *death tax* instead of *estate tax*). Moreover, it is easy and efficient to represent texts as collections of the words they contain, in order to apply a well known arsenal of supervised techniques (Laver et al., 2003; Mullen and Malouf, 2006; Yu et al., 2008).

At the same time, standard lexical features have their limitations for this kind of analysis. Such features are usually created by selecting some small n-gram size in advance. Indeed, it is not uncommon to see the feature space for sentiment analysis limited to unigrams. However, important indicators of perspective can also be longer (*get the government out of my*). Trying to capture these using standard machine learning approaches creates a problem, since allowing n-grams as features for larger n gives rise to problems of data sparsity.

In this paper, we employ nonparametric Bayesian models (Orbanz and Teh, 2010) in order to address this limitation. In contrast to parametric models, for which a fixed number of parameters are specified in advance, nonparametric models can "grow" to the size best suited to the observed data. In text analysis, models of this type have been employed primarily for unsupervised discovery of latent structure — for example, in topic modeling, when the true number of topics is not known (Teh et al., 2006); in grammatical inference, when the appropriate number of nonterminal symbols is not known (Liang et al., 2007); and in coreference resolution, when the number of entities in a given document is not specified in advance (Haghighi and Klein, 2007). Here we use them for supervised text classification.

Specifically, we use *adaptor grammars* (Johnson et al., 2006), a formalism for nonparametric Bayesian modeling that has recently proven useful in unsupervised modeling of phonemes (Johnson, 2008), grammar induction (Cohen et al., 2010), and named entity structure learning (Johnson, 2010), to make supervised naïve Bayes classification nonparametric in order to improve perspective modeling. Intuitively, naïve Bayes associates each class or label with a probability distribution over a fixed vocabulary. We introduce *adaptive naïve Bayes* (ANB), for which in principle the vocabulary can grow as needed to include collocations of arbitrary length, as determined

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 284–292,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

by the properties of the dataset. We show that using adaptive naïve Bayes improves on state of the art classification using the Bitter Lemons corpus (Lin et al., 2006), a document collection that has been used by a variety of authors to evaluate perspective classification.

In Section 2, we review adaptor grammars, show how naïve Bayes can be expressed within the formalism, and describe how — and how easily — an adaptive naïve Bayes model can be created. Section 3 validates the approach via experimentation on the Bitter Lemons corpus. In Section 4, we summarize the contributions of the paper and discuss directions for future work.

2 Adapting Naïve Bayes to be Less Naïve

In this work we apply the *adaptor grammar* formalism introduced by Johnson, Griffiths, and Goldwater (Johnson et al., 2006). Adaptor grammars are a generalization of probabilistic context free grammars (PCFGs) that make it particularly easy to express nonparametric Bayesian models of language simply and readably using context free rules. Moreover, Johnson *et al.* provide an inference procedure based on Markov Chain Monte Carlo techniques that makes parameter estimation straightforward for all models that can be expressed using adaptor grammars.[1] Variational inference for adaptor grammars has also been recently introduced (Cohen et al., 2010).

Briefly, adaptor grammars allow nonterminals to be rewritten to entire subtrees. In contrast, a nonterminal in a PCFG rewrites only to a collection of grammar symbols; their subsequent productions are independent of each other. For instance, a traditional PCFG might learn probabilities for the rewrite rule PP $\mapsto$ P NP. In contrast, an adaptor grammar can learn (or "cache") the production PP $\mapsto$ (P up)(NP(DET a)(N tree)). It does this by positing that the distribution over children for an adapted non-terminal comes from a Pitman-Yor distribution.

A Pitman-Yor distribution (Pitman and Yor, 1997) is a distribution over distributions. It has three parameters: the discount, a, such that $0 \leq a < 1$, the strength, b, a real number such that $-a < b$, and a probability distribution G_0 known as the base distribution. Adaptor grammars allow distributions over subtrees to come from a Pitman-Yor distribution with the PCFG's original distribution over trees as the base distribution. The generative process for obtaining draws from a distribution drawn from a Pitman-Yor distribution can be described by the "Chinese restaurant process" (CRP). We will use the CRP to describe how to obtain a distribution over observations composed of sequences of n-grams, the key to our model's ability to capture perspective-bearing n-grams.

Suppose that we have a base distribution Ω that is some distribution over all sequences of words (the exact structure of such a distribution is unimportant; such a distribution will be defined later in Table 1). Suppose further we have a distribution ϕ drawn from $PY(a, b, \Omega)$, and we wish to draw a series of observations $\boldsymbol{w}$ from ϕ. The CRP gives us a generative process for doing those draws from ϕ, marginalizing out ϕ. Following the restaurant metaphor, we imagine the i^{th} customer in the series entering the restaurant to take a seat at a table. The customer sits by making a choice that determines the value of the n-gram w_i for that customer: she can either sit at an existing table or start a new table of her own.[2]

If she sits at a new table j, that table is assigned a draw y_j from the base distribution, Ω; note that, since Ω is a distribution over n-grams, y_j is an n-gram. The value of w_i is therefore assigned to be y_j, and y_j becomes the sequence of words assigned to that new table. On the other hand, if she sits at an existing table, then w_i simply takes the sequence of words already associated with that table (assigned as above when it was first occupied).

The probability of joining an existing table j, with c_j patrons already seated at table j, is $\frac{c_j - a}{c. + b}$, where $c.$ is the number of patrons seated at all tables: $c. = \sum_{j'} c_{j'}$. The probability of starting a new table is $\frac{b + t * a}{c. + b}$, where t is the number of tables presently occupied.

Notice that ϕ is a distribution over the same space as Ω, but it can drastically shift the mass of the distribution, compared with Ω, as more and more pa-

[1] And, better still, they provide code that implements the inference algorithm; see http://www.cog.brown.edu/ mj/Software.htm.

[2] Note that we are abusing notation by allowing w_i to correspond to a word sequence of length ≥ 1 rather than a single word.

trons are seated at tables. However, there is always a chance of drawing from the base distribution, and therefore every word sequence can also always be drawn from ϕ.

In the next section we will write a naïve Bayes-like generative process using PCFGs. We will then use the PCFG distribution as the base distribution for a Pitman-Yor distribution, adapting the naïve Bayes process to give us a distribution over n-grams, thus learning new language substructures that are useful for modeling the differences in perspective.

2.1 Classification Models as PCFGs

Naïve Bayes is a venerable and popular mechanism for text classification (Lewis, 1998). It posits that there are K distinct categories of text — each with a distinct distribution over words — and that every document, represented as an exchangeable bag of words, is drawn from one (and only one) of these distributions. Learning the per-category word distributions and global prevalence of the classes is a problem of posterior inference which can be approached using a variety of inference techniques (Lowd and Domingos, 2005).

More formally, naïve Bayes models can be expressed via the following generative process:[3]

1. Draw a global distribution over classes $\theta \sim \text{Dir}(\alpha)$
2. For each class $i \in \{1, \dots, K\}$, draw a word distribution $\phi_i \sim \text{Dir}(\lambda)$
3. For each document $d \in \{1, \dots, M\}$:
   (a) Draw a class assignment $z_d \sim \text{Mult}(\theta)$
   (b) For each word position $n \in \{1, \dots, N_d$, draw $w_{d,n} \sim \text{Mult}(\phi_{z_d})$

A variant of the naïve Bayes generative process can be expressed using the adaptor grammar formalism (Table 1). The left hand side of each rule represents a nonterminal which can be expanded, and the right hand side represents the rewrite rule. The rightmost indices show replication; for instance, there are $|V|$ rules that allow Word$_i$ to rewrite to each word in the vocabulary. One can assume a symmetric Dirichlet prior of $\text{Dir}(\bar{1})$ over the production choices unless otherwise specified — as with the Doc$_d$ production rule above, where a sparse prior is used.

Sent	$\mapsto$	Doc$_d$	$d = 1, \dots, m$
Doc$_d{}^{0.001}$	$\mapsto$	ID$_d$ Words$_i$	$d = 1, \dots, m;$ $i \in \{1, K\}$
Words$_i$	$\mapsto$	Words$_i$ Word$_i$	$i \in \{1, K\}$
Words$_i$	$\mapsto$	Word$_i$	$i \in \{1, K\}$
Word$_i$	$\mapsto$	v	$v \in V; i \in \{1, K\}$

Table 1: A naïve Bayes-inspired model expressed as a PCFG.

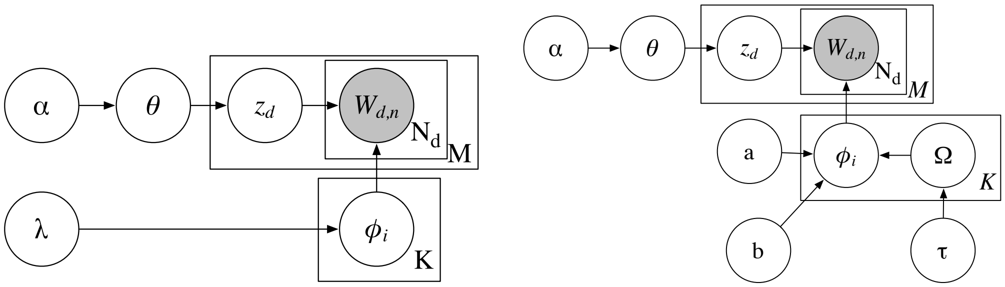
Notice that the distribution over expansions for Word$_i$ corresponds directly to ϕ_i in Figure 1(a). There are, however, some differences between the model that we have described above and the standard naïve Bayes model depicted in Figure 1(a). In particular, there is no longer a single choice of class per document; each *sentence* is assigned a class. If the distribution over per-sentence labels is sparse (as it is above for Doc$_d$), this will closely approximate naïve Bayes, since it will be very unlikely for the sentences in a document to have different labels. A non-sparse prior leads to behavior more like models that allow parts of a document to express sentiment or perspective differently.

2.2 Moving Beyond the Bag of Words

The naïve Bayes generative distribution posits that when writing a document, the author selects a distribution of categories z_d for the document from θ. The author then generates words one at a time: each word is selected independently from a flat multinomial distribution ϕ_{z_d} over the vocabulary.

However, this is a very limited picture of how text is related to underlying perspectives. Clearly words are often connected with each other as collocations, and, just as clearly, extending a flat vocabulary to include bigram collocations does not suffice, since sometimes relevant perspective-bearing phrases are longer than two words. Consider phrases like *health care for all* or *government takeover of health care*, connected with progressive and conservative positions, respectively, during the national debate on healthcare reform. Simply applying naïve Bayes, or any other model, to a bag of n-grams for high n is

[3] Here α and λ are hyperparameters used to specify priors for the class distribution and classes' word distributions, respectively; α is a symmetric K-dimensional vector where each element is π. N_d is the length of document d. Resnik and Hardisty (2010) provide a tutorial introduction to the naïve Bayes generative process and underlying concepts.



(a) Naïve Bayes

(b) Adaptive Naïve Bayes

Figure 1: A plate diagram for naïve Bayes and adaptive naïve Bayes. Nodes represent random variables and parameters; shaded nodes represent observations; lines represent probabilistic dependencies; and the rectangular plates denote replication.

going to lead to unworkable levels of data sparsity; a model should be flexible enough to support both unigrams and longer phrases as needed.

Following Johnson (2010), however, we can use adaptor grammars to extend naïve Bayes *flexibly* to include richer structure like collocations when they improve the model, and not including them when they do not. This can be accomplished by introducing *adapted* nonterminal rules: in a revised generative process, the author can draw from Pitman-Yor distribution whose base distribution is over word *sequences* of arbitrary length.[4] Thus in a setting where, say, $K = 2$, and our two classes are PROGRESSIVE and CONSERVATIVE, the sequence *health care for all* might be generated as a single unit for the progressive perspective, but in the conservative perspective the same sequence might be generated as three separate draws: *health care*, *for*, *all*. Such a model is presented in Figure 1(b). Note the following differences between Figures 1(a) and 1(b):

- z_d selects which Pitman-Yor distribution to draw from for document d.
- ϕ_i is the distribution over n-grams that comes from the Pitman-Yor distribution.
- $W_{d,n}$ represents an n-gram draw from ϕ_i
- a, b are the Pitman-Yor strength and discount parameters.
- Ω is the Pitman-Yor base distribution with τ as its uniform hyperparameter.

Returning to the CRP metaphor discussed when we introduced the Pitman-Yor distribution, there are two restaurants, one for the PROGRESSIVE distribution and one for the CONSERVATIVE distribution. *Health care for all* has its own table in the PROGRESSIVE restaurant, and enough people are sitting at it to make it popular. There is no such table in the CONSERVATIVE restaurant, so in order to generate those words, the phrase *health care for all* would need to come from a new table; however, it is more easily explained by three customers sitting at three existing, popular tables: *health care*, *for*, and *all*.

We follow the convention of Johnson (2010) by writing adapted nonterminals as underlined. The grammar for adaptive naïve Bayes is shown in Table 2. The adapted $\underline{\text{COLLOC}_i}$ rule means that every time we need to generate that nonterminal, we are actually drawing from a distribution drawn from a Pitman-Yor distribution. The distribution over the possible yields of the WORDS$_i$ rule serves as the base distribution.

Given this generative process for documents, we can now use statistical inference to uncover the posterior distribution over the latent variables, thus discovering the tables and seating assignments of our metaphorical restaurants that each cater to a specific perspective filled with tables populated by words and n-grams.

The model presented in Table 2 is the most straightforward way of extending naïve Bayes to collocations. For completeness, we also consider the alternative of using a shared base distribution rather than distinguishing the base distributions of the two classes.

[4] As defined above, the base distribution is that of the PCFG production rule WORDS$_i$. Although it has non-zero probability of producing any sequence of words, it is biased toward shorter word sequences.

SENT	$\mapsto$	DOC$_d$	$d = 1, \ldots, m$
DOC$_d^{0.001}$	$\mapsto$	ID$_d$ SPAN$_i$	$d = 1, \ldots, m;$ $i \in \{1, K\}$
SPAN$_i$	$\mapsto$	SPAN$_i$ COLLOC$_i$	$i \in \{1, K\}$
SPAN$_i$	$\mapsto$	COLLOC$_i$	$i \in \{1, K\}$
$\underline{\text{COLLOC}_i}$	$\mapsto$	WORDS$_i$	$i \in \{1, K\}$
WORDS$_i$	$\mapsto$	WORDS$_i$ WORD$_i$	$i \in \{1, K\}$
WORDS$_i$	$\mapsto$	WORD$_i$	$i \in \{1, K\}$
WORD$_i$	$\mapsto$	v	$v \in V; i \in \{1, K\}$

Table 2: An adaptive naïve Bayes grammar. The $\underline{\text{COLLOC}_i}$ nonterminal's distribution over yields is drawn from a Pitman-Yor distribution rather than a Dirichlet over production rules.

SENT	$\mapsto$	DOC$_d$	$d = 1, \ldots, m$
DOC$_d^{0.001}$	$\mapsto$	ID$_d$ SPAN$_i$	$d = 1, \ldots, m;$ $i \in \{1, K\}$
SPAN$_i$	$\mapsto$	SPAN$_i$ COLLOC$_i$	$i \in \{1, K\}$
SPAN$_i$	$\mapsto$	COLLOC$_i$	$i \in \{1, K\}$
$\underline{\text{COLLOC}_i}$	$\mapsto$	WORDS	$i \in \{1, K\}$
WORDS	$\mapsto$	WORDS WORD	
WORDS	$\mapsto$	WORD	
WORD	$\mapsto$	v	$v \in V$

Table 3: An adaptive naïve Bayes grammar with a common base distribution for collocations. Note that, in contrast to Table 2, there are no subscripts on WORDS or WORD.

Briefly, using a shared base distribution posits that the two classes use similar word distributions, but generate collocations unique to each class, whereas using separate base distributions assumes that the distribution of words is unique to each class.

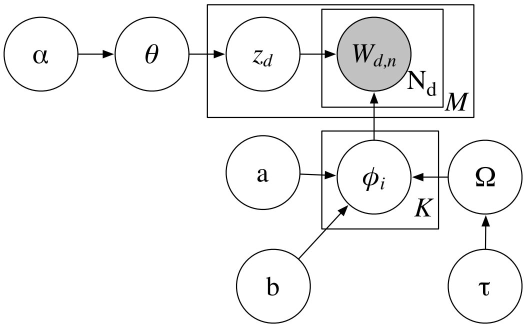


Figure 2: An alternative adaptive naïve Bayes with a common base distribution for both classes.

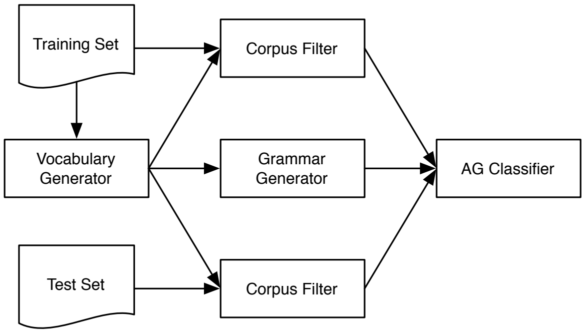


Figure 3: Corpus preparation and experimental setup.

3 Experiments

3.1 Corpus Description

We conducted our classification experiments on the Bitter Lemons (BL) corpus, which is a collection of 297 essays averaging 700-800 words in length, on various Middle East issues, written from both the Israeli and Palestinian perspectives. The BL corpus was compiled by Lin *et al.* (2006) and is derived from a website that invites weekly discussions on a topic and publishes essays from two sets of authors each week.[5] Two of the authors are guests, one from each perspective, and two essays are from the site's regular contributors, also one from each perspective, for a total of four essays on each topic per week. We chose this corpus to allow us to directly compare our results with Greene and Resnik's (2009) Observable Proxies for Underlying Semantics (OPUS) features and Lin *et al.*'s Latent Sentence Perspective Model (LSPM). The classification goal for this corpus is to label each document with the perspective of its author, either Israeli or Palestinian.

Consistent with prior work, we prepared the corpus by dividing it into two groups, one group containing all of the essays written by the regular site contributors, which we call the Editor set, and one group comprised of all the essays written by the guest contributors, which we call the Guest set. Similar to the above mentioned prior work, we perform classification using one group as training data and the other as test data and perform two folds of classification. The overall experimental setup and corpus preparation process is presented in Figure 3.

[5] `http://www.bitterlemons.org`

3.2 Experimental Setup

The vocabulary generator determines the vocabulary used by a given experiment by converting the training set to lower case, stemming with the Porter stemmer, and filtering punctuation. We remove from the vocabulary any words that appeared in only one document regardless of frequency within that document, words with frequencies lower than a threshold, and stop words.[6] The vocabulary is then passed to a grammar generator and a corpus filter.

The grammar generator uses the vocabulary to generate the terminating rules of the grammar from the ANB grammar presented in Tables 2 and 3. The corpus filter takes in a set of documents and replaces all words not in the vocabulary with "out of vocabulary" markers. This process ensures that in all experiments the vocabulary is composed entirely of words from the training set. After the groups have been filtered, the group used as the test set has its labels removed. The test and training set are then sent, along with the grammar, into the adaptor grammar inference engine.

Each experiment ran for 3000 iterations. For the runs where adaptation was used we set the initial Pitman-Yor a and b parameters to 0.01 and 10 respectively, then slice sample (Johnson and Goldwater, 2009).

We use the resulting sentence parses for classification. By design of the grammar, each sentence's words will belong to one and only one distribution. We identify that distribution from each of the test set sentence parses and use it as the sentence level classification for that particular sentence. We then use majority rule on the individual sentence classifications in a document to obtain the document classification. (In most cases the sentence-level assignments are overwhelmingly dominated by one class.)

3.3 Results and Analysis

Table 4 gives the results and compares to prior work. The support vector machine (SVM), NB-B and LSPM results are taken directly from Lin *et al.* (2006). NB-B indicates naïve Bayes with full Bayesian inference. LSPM is the Latent Sentence Perspective Model, also from Lin *et al.* (2006). OPUS results are taken from Greene and Resnik (2009). Briefly, OPUS features are generated from observable grammatical relations that come from dependency parses of the corpus. Use of these features provided the best classification accuracy for this task prior to this work. ANB* refers to the grammar from Table 2, but with adaptation disabled. The reported accuracy values for ANB*, ANB with a common base distribution (see Table 3), and ANB with separate base distributions (see Table 2) are the mean values from five separate sampling chains. Bold face indicates statistical signficance ($p < 0.05$) by unpaired t-test between the reported value and ANB*.

Training Set	Test Set	Classifier	Accuracy
Guests	Editors	SVM	88.22
Guests	Editors	NB-B	93.46
Guests	Editors	LSPM	94.93
Guests	Editors	OPUS	97.64
Guests	Editors	ANB*	99.32
Guests	Editors	ANB Com	**99.93**
Guests	Editors	ANB Sep	**99.87**
Editors	Guests	SVM	81.48
Editors	Guests	NB-B	85.85
Editors	Guests	LSPM	86.99
Editors	Guests	OPUS	85.86
Editors	Guests	ANB*	84.98
Editors	Guests	ANB Com	82.76
Editors	Guests	ANB Sep	**88.28**

Table 4: Classification results. ANB* indicates the same grammar as Adapted Naïve Bayes, but with adaptation disabled. Com and Sep refer to whether the base distribution was common to both classes or separate.

Consistent with all prior work on this corpus we found that the classification accuracy for training on editors and testing on guests was lower than the other direction since the larger number of editors in the guest set allows for greater generalization. The difference between ANB* and ANB with a common base distribution is not statistically significant. Also of note is that the classification accuracy improves for testing on Guests when the ANB grammar is allowed to adapt and a separate base distribution is used for the two classes (88.28% versus 84.98% without adaptation).

Table 5 presents some data on adapted rules

[6] In these experiments, a frequency threshold of 4 was selected prior to testing.

Class	Group	Unique Unigrams Cached	Unique n-grams Cached	Percent of Group Vocabulary Cached
Israeli	Editors	2,292	19,614	77.62
Palestinian	Editors	2,180	17,314	86.54
Israeli	Guests	2,262	19,398	79.91
Palestinian	Guests	2,005	16,946	74.94

Table 5: Counts of cached unigrams and n-grams for the two classes compared to the vocabulary sizes.

Israeli	Palestinian
zionist dream	american jew
zionist state	achieve freedom
zionist movement	palestinian freedom
american leadership	support palestinian
american victory	palestinian suffer
abandon violence	palestinian territory
freedom (of the) press	palestinian statehood
palestinian violence	palestinian refugee

Table 6: Charged bigrams captured by the framework.

learned once inference is complete. The column labeled *unique unigrams cached* indicates the number of unique unigrams that appear on the right hand side of the adapted rules. Similarly, *unique n-grams cached* indicates the number of unique n-grams that appear on the right hand side of the adapted rules. The rightmost column indicates the percentage of terms from the group vocabulary that appear on the right hand side of adapted rules as unigrams. Values less than 100% indicate that the remaining vocabulary terms are cached in n-grams. As the table shows, a significant number of the rules learned during inference are n-grams of various sizes.

Inspection of the captured bigrams showed that it captured sequences that a human might associate with one perspective over the other. Table 6 lists just a few of the more charged bigrams that were captured in the adapted rules.

More specific caching information on the individual groups and classes is provided in Table 7. This data clearly demonstrates that raw n-gram frequency alone is not indicative of how many times an n-gram is used as a cached rule. For example, consider the bigram *people go*, which is used as a cached bigram only three times, yet appears in the corpus 407 times. Compare that with *isra palestinian*, which is cached the same number of times but appears only 18 times in the corpus. In other words, the sequence *people go* is more easily explained by two sequential unigrams, not a bigram. The ratio of cache use counts to raw bigrams gives a measure of strength of collocation between the terms of the n-gram. We conjecture that the rareness of caching for $n > 2$ is a function of the small corpus size. Also of note is the improvement in performance of ANB* over NB-B when training on guests, which we suspect is due to our use of sampled versus fixed hyperparameters.

4 Conclusions

In this paper, we have applied adaptor grammars in a supervised setting to model lexical properties of text and improve document classification according to perspective, by allowing nonparametric discovery of collocations that aid in perspective classification. The adaptive naïve Bayes model improves on state of the art supervised classification performance in head-to-head comparisons with previous approaches.

Although there have been many investigations on the efficacy of using multiword collocations in text classification (Bekkerman and Allan, 2004), usually such approaches depend on a preprocessing step such as computing *tf-idf* or other measures of frequency based on either word bigrams (Tan et al., 2002) or character n-grams (Raskutti et al., 2001). In contrast, our approach combines phrase discovery with the probabilistic model of the text. This allows for the collocation selection and classification to be expressed in a single model, which can then be extended later; it also is truly generative, as compared with feature induction and selection algorithms that either under- or over-generate data.

There are a number of interesting directions in which to take this research. As Johnson *et al.* (2006) argue, and as we have confirmed here, the adaptor

Guest						Editor					
Israeli			Palestinian			Israeli			Palestinian		
palestinian OOV	11	299	palestinian isra	6	178	palestinian OOV	8	254	OOV israel	7	198
OOV palestinian	7	405	OOV palestinian	6	405	OOV palestinian	7	319	OOV palestinian	6	319
isra OOV	6	178	palestinian OOV	5	29	OOV israel	7	123	OOV work	5	254
israel OOV	6	94	one OOV	4	25	OOV us	6	115	OOV agreement	5	75
sharon OOV	4	74	side OOV	3	21	OOV part	5	56	palestinian reform	4	49
polit OOV	4	143	polit OOV	3	299	israel OOV	5	81	palestinian OOV	4	81
OOV us	4	29	peopl go	3	407	attempt OOV	5	91	OOV isra	4	15
OOV state	4	37	palestinian govern	3	94	time OOV	4	63	one OOV	4	27
israel palestinian	4	52	palestinian accept	3	220	remain OOV	4	85	isra palestinian	4	17
even OOV	4	43	OOV state	3	150	OOV time	4	70	isra OOV	4	63
arafat OOV	4	41	OOV israel	3	18	OOV area	4	49	howev OOV	4	149
appear OOV	4	53	OOV end	3	20	OOV arafat	4	28	want OOV	3	36
total OOV	3	150	OOV act	3	105	isra OOV	4	8	us OOV	3	35
palestinian would	3	65	isra palestinian	3	18	would OOV	3	28	recent OOV	3	220
palestinian isra	3	35	israel OOV	3	198	use OOV	3	198	palestinian isra	3	115

Table 7: Most frequently used cached bigrams. The first colum in each section is the number of times that bigram was used as a cached rule. The second column indicates the raw count of that bigram in the Guests or Editors group.

grammar formalism makes it quite easy to work with latent variable models, in order to automatically discover structures in the data that have predictive value. For example, it is easy to imagine a model where in addition to a word distribution for each class, there is also an additional shared "neutral" distribution: for each sentence, the words in that sentence can either come from the class-specific content distribution or the shared neutral distribution. This turns out to be the Latent Sentence Perspective Model of Lin *et al.* (2006), which is straightforward to encode using the adaptor grammar formalism simply by introducing two new nonterminals to represent the neutral distribution:

$$\begin{array}{llll}
\textsc{Sent} & \mapsto & \textsc{Doc}_d & d = 1, \ldots, m \\
\textsc{Doc}_d & \mapsto & \textsc{ID}_d\ \textsc{Words}_i & d = 1, \ldots, m; \\
 & & & i \in \{1, K\} \\
\textsc{Doc}_d & \mapsto & \textsc{ID}_d\ \textsc{Neuts} & d = 1, \ldots, m; \\
\textsc{Words}_i & \mapsto & \textsc{Words}_i\ \textsc{Word}_i & i \in \{1, K\} \\
\textsc{Words}_i & \mapsto & \textsc{Word}_i & i \in \{1, K\} \\
\textsc{Word}_i & \mapsto & \text{v} & v \in V; i \in \{1, K\} \\
\textsc{Neut} & \mapsto & \textsc{Neuts}_i\ \textsc{Neut}_i & \\
\textsc{Neut} & \mapsto & \textsc{Neut} & \\
\textsc{Neut} & \mapsto & \text{v} & v \in V
\end{array}$$

Running this grammar did not produce improvements consistent with those reported by Lin *et al.* We plan to investigate this further, and a natural follow-on would be to experiment with adaptation for this variety of latent structure, to produce an adapted LSPM-like model analogous to adaptive naïve Bayes.

Viewed in a larger context, computational classification of perspective is closely connected to social scientists' study of *framing*, which Entman (1993) characterizes as follows: "To frame is to select some aspects of a perceived reality and make them more salient in a communicating text, in such a way as to promote a particular problem definition, causal interpretation, moral evaluation, and/or treatment recommendation for the item described." Here and in other work (e.g. (Laver et al., 2003; Mullen and Malouf, 2006; Yu et al., 2008; Monroe et al., 2008)), it is clear that lexical evidence is one key to understanding how language is used to frame discussion from one perspective or another; Resnik and Greene (2009) have shown that syntactic choices can provide important evidence, as well. Another promising direction for this work is the application of adaptor grammar models as a way to capture both lexical and grammatical aspects of framing in a unified model.

Acknowledgments

This research was funded in part by the Army Research Laboratory through ARL Cooperative Agreement W911NF-09-2-0072 and by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), through the Army Research Laboratory. All statements of fact, opinion or conclusions contained herein are those of the authors and should not be construed as representing the official views or policies of ARL, IARPA, the ODNI, or the U.S. Government. The authors thank Mark Johnson and the anonymous reviewers for their helpful comments and discussions. We are particularly grateful to Mark Johnson for making his adaptor grammar code available.

References

R. Bekkerman and J. Allan. 2004. Using bigrams in text categorization. Technical Report IR-408, Center of Intelligent Information Retrieval, UMass Amherst.

S. B. Cohen, D. M. Blei, and N. A. Smith. 2010. Variational inference for adaptor grammars. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

R.M. Entman. 1993. Framing: Toward Clarification of a Fractured Paradigm. *The Journal of Communication*, 43(4):51–58.

Stephan Greene and Philip Resnik. 2009. More than words: Syntactic packaging and implicit sentiment. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 503–511.

Aria Haghighi and Dan Klein. 2007. Unsupervised coreference resolution in a nonparametric bayesian model. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 848–855, Prague, Czech Republic, June.

Mark Johnson and Sharon Goldwater. 2009. Improving nonparameteric bayesian inference: experiments on unsupervised word segmentation with adaptor grammars. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 317–325, Boulder, Colorado, June.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2006. Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. In *Proceedings of Advances in Neural Information Processing Systems*.

Mark Johnson. 2008. Using adaptor grammars to identify synergies in the unsupervised acquisition of linguistic structure. In *Proceedings of ACL-08: HLT*, pages 398–406, Columbus, Ohio, June.

Mark Johnson. 2010. PCFGs, topic models, adaptor grammars and learning topical collocations and the structure of proper names. In *Proceedings of the Association for Computational Linguistics*, pages 1148–1157, Uppsala, Sweden, July.

Michael Laver, Kenneth Benoit, and John Garry. 2003. Extracting policy positions from political texts using words as data. *American Political Science Review*, pages 311–331.

David D. Lewis. 1998. Naive (bayes) at forty: The independence assumption in information retrieval. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 4–15, Chemnitz, DE. Springer Verlag, Heidelberg, DE.

Percy Liang, Slav Petrov, Michael Jordan, and Dan Klein. 2007. The infinite PCFG using hierarchical Dirichlet processes. In *Proceedings of Emperical Methods in Natural Language Processing*, pages 688–697.

Wei-Hao Lin, Theresa Wilson, Janyce Wiebe, and Alexander Hauptmann. 2006. Which side are you on? Identifying perspectives at the document and sentence levels. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*.

Daniel Lowd and Pedro Domingos. 2005. Naive bayes models for probability estimation. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 529–536, New York, NY, USA. ACM.

Burt L. Monroe, Michael P. Colaresi, and Kevin M. Quinn. 2008. Fightin' Words: Lexical Feature Selection and Evaluation for Identifying the Content of Political Conflict. *Political Analysis, Vol. 16, Issue 4, pp. 372-403, 2008*.

Tony Mullen and Robert Malouf. 2006. A preliminary investigation into sentiment analysis of informal political discourse. In *AAAI Symposium on Computational Approaches to Analysing Weblogs (AAAI-CAAW)*, pages 159–162.

P. Orbanz and Y. W. Teh. 2010. Bayesian nonparametric models. In *Encyclopedia of Machine Learning*. Springer.

J. Pitman and M. Yor. 1997. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25(2):855–900.

Bhavani Raskutti, Herman L. Ferrá, and Adam Kowalczyk. 2001. Second order features for maximising text classification performance. In *EMCL '01: Proceedings of the 12th European Conference on Machine Learning*, pages 419–430, London, UK. Springer-Verlag.

Philip Resnik and Eric Hardisty. 2010. Gibbs sampling for the uninitiated. Technical Report UMIACS-TR-2010-04, University of Maryland. http://www.lib.umd.edu/drum/handle/1903/10058.

Chade-Meng Tan, Yuan-Fang Wang, and Chan-Do Lee. 2002. The use of bigrams to enhance text categorization. *Inf. Process. Manage.*, 38(4):529–546.

Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

B. Yu, S. Kaufmann, and D. Diermeier. 2008. Classifying party affiliation from political speech. *Journal of Information Technology and Politics*, 5(1):33–48.

Predicting the Semantic Compositionality of Prefix Verbs

Shane Bergsma, Aditya Bhargava, Hua He, Grzegorz Kondrak
Department of Computing Science
University of Alberta
{bergsma,abhargava,hhe,kondrak}@cs.ualberta.ca

Abstract

In many applications, replacing a complex word form by its stem can reduce sparsity, revealing connections in the data that would not otherwise be apparent. In this paper, we focus on prefix verbs: verbs formed by adding a prefix to an existing verb stem. A prefix verb is considered compositional if it can be decomposed into a semantically equivalent expression involving its stem. We develop a classifier to predict compositionality via a range of lexical and distributional features, including novel features derived from web-scale N-gram data. Results on a new annotated corpus show that prefix verb compositionality can be predicted with high accuracy. Our system also performs well when trained and tested on conventional morphological segmentations of prefix verbs.

1 Introduction

Many verbs are formed by adding prefixes to existing verbs. For example, *remarry* is composed of a prefix, *re*-, and a stem, *marry*. We present an approach to predicting the compositionality of prefix verbs. The verb *remarry* is compositional; it means to *marry again*. On the other hand, *retire* is generally non-compositional; it rarely means *to tire again*. There is a continuum of compositionality in prefix verbs, as in other complex word forms and multiword expressions (Bannard et al., 2003; Creutz and Lagus, 2005; Fazly et al., 2009; Xu et al., 2009).

We adopt a definition of compositionality specifically designed to support downstream applications that might benefit from knowledge of verb stems. For example, suppose our corpus contains the following sentence: “Pope Clement VII denied Henry VIII permission to marry again before a decision was given in Rome.” A user might submit the question, “Which pope refused Henry VIII permission to remarry?” If we can determine that the meaning of *remarry* could also be provided via the stem *marry*, we could add *marry* to our search terms. This is known as *morphological query expansion* (Bilotti et al., 2004). Here, such an expansion leads to a better match between question and answer.

Previous work has shown that “full morphological analysis provides at most very modest benefits for retrieval” (Manning et al., 2008). Stemming, lemmatization, and compound-splitting often increase recall at the expense of precision, but the results depend on the morphological complexity of the text’s language (Hollink et al., 2004).

The lack of success in applying morphological analysis in IR is unsurprising given that most previous systems are not designed with applications in mind. For example, the objective of the influential *Linguistica* program is “to produce an output that matches as closely as possible the analysis that would be given by a human morphologist” (Goldsmith, 2001). Unsupervised systems achieve this aim by exploiting learning biases such as minimum description length for lexicons (Goldsmith, 2001; Creutz and Lagus, 2007) and high entropy across morpheme boundaries (Keshava and Pitler, 2006). Supervised approaches learn directly from words annotated by morphologists (Van den Bosch and Daelemans, 1999; Toutanova and Cherry, 2009), often using CELEX, a lexical database that includes

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 293–303,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

morphological information (Baayen et al., 1996).

The conventional approach in morphology is to segment words into separate morphemes even when the words are not entirely compositional combinations of their parts (Creutz and Lagus, 2005). For example, while *co-* is considered a separate morpheme in the verb *cooperate*, the meaning of *cooperate* is not simply *to operate jointly*. These forms are sometimes viewed as *perturbations* of composition (de Marken, 1996). In practice, a user may query, "Which nations do not cooperate with the International Criminal Court?" An expansion of the query to include *operate* may have undesirable consequences.

Rather than relying on conventional standards, we present an algorithm whose objective is to find only those prefix verbs that exhibit semantic compositionality; i.e., prefix verbs that are fully meaning-preserving, sums-of-their-parts. We produce a new corpus, annotated according to this definition. We use these annotated examples to learn a discriminative model of semantic compositionality.

Our classifier relies on a variety of features that exploit the distributional patterns of verbs and stems. We build on previous work that applies semantics to morphology (Yarowsky and Wicentowski, 2000; Schone and Jurafsky, 2001; Baroni et al., 2002), and also on work that exploits web-scale data for semantic analysis (Turney, 2001; Nakov, 2007; Kummerfeld and Curran, 2008). For example, we measure how often a prefix verb appears with a hyphen between the prefix and stem. We also look at the distribution of the stem as a separate word: we calculate the probability of the prefix verb and the separated stem's co-occurrence in a segment of discourse; we also calculate the distributional similarity between the verb and the separated stem. High scores for these measures indicate compositionality. We extract counts from a web-scale N-gram corpus, allowing us to efficiently leverage huge volumes of unlabeled text.

Our system achieves 93.6% accuracy on held-out data, well above several baselines and comparison systems. We also train and test our system on conventional morphological segmentations. Our classifier remains reliable in this setting, making half as many errors as the state-of-the-art unsupervised Morfessor system (Creutz and Lagus, 2007).

2 Problem Definition and Setting

A prefix verb is a derived word with a bound morpheme as prefix. While derivation can change both the meaning and part-of-speech of a word (as opposed to inflection, which does not change "referential or cognitive meaning" (Katamba, 1993)), here the derived form remains a verb.

We define prefix-verb compositionality as a semantic equivalence between a verb and a paraphrase involving the verb's stem. The stem must be used as a verb in the paraphrase. Words can be introduced, if needed, to account for the meaning contributed by the prefix, e.g., *outbuild⇒build more/better/faster than*. A bidirectional entailment between the prefix verb and the paraphrase is required.

Words can have different meanings in different contexts. For example, a nation might "*resort* to force," (non-compositional) while a computer program can "*resort* a linked list" (compositional). We therefore define prefix-verb compositionality as a context-specific property of verb tokens rather than a global property of verb types. However, it is worth noting that we ultimately found the compositionality of types to be very consistent across contexts (Section 5.1.2), and we were unable to leverage contextual information to improve classification accuracy; our final system is essentially type-based. Other recent morphological analyzers have also been type-based (Keshava and Pitler, 2006; Poon et al., 2009).

Our system takes as input a verb token in uninflected form along with its sentence as context. The verb must be divisible into an initial string and a following remainder such that the initial string is on our list of prefixes and the remainder is on our list of stems. Hyphenation is allowed, e.g., both *re-enter* and *reenter* are acceptable inputs. The system determines whether the prefix/stem combination is compositional in the current context. For example, the verb *unionize* in, "The workers must unionize," can be divided into a prefix *un-* and a stem *ionize*. The system should determine that here *unionize* is not a compositional combination of these parts.

The algorithm requires a list of prefixes and stems in a given language. For our experiments, we use both dictionary and corpus-based methods to construct these lists (Section 4).

3 Supervised Compositionality Detection

We use a variety of lexical and statistical information when deciding whether a prefix verb is compositional. We adopt a discriminative approach. We assume some labeled examples are available to train a classifier. Relevant information is encoded in a feature vector, and a learning algorithm determines a set of weights for the features using the training data. As compositionality is a binary decision, we can adopt any standard package for binary classification. In our experiments we use support vector machines.

Our features include both local information that depends only on the verb string (sometimes referred to as lexical features) and also global information that depends on the verb and the stem's distribution in text. Our approach can therefore be regarded as a simple form of semi-supervised learning; we leverage both a small number of labeled examples and a large volume of unlabeled text.

If a frequency or similarity is undefined in our corpus, we indicate this with a separate feature; weights on these features act as a kind of smoothing.

3.1 Features based on Web-Scale N-gram Data

We use web-scale N-gram data to extract distributional features. The most widely-used N-gram corpus is the Google 5-gram Corpus (Brants and Franz, 2006). We use *Google V2*: a new N-gram corpus (also with N-grams of length one-to-five) created from the same one-trillion-word snapshot of the web as the Google 5-gram Corpus, but with enhanced filtering and processing of the source text (Lin et al., 2010). For Google V2, the source text was also part-of-speech tagged, and the resulting part-of-speech tag distribution is included for each N-gram. There are 4.1 billion N-grams in the corpus.

The part-of-speech tag distributions are particularly useful, as they allow us to collect verb-specific counts. For example, while a string like *reuse* occurs 1.1 million times in the web corpus, it is only tagged as a verb 270 thousand times. Conflating the noun/verb senses can lead to misleading scores for certain features. E.g., the hyphenation frequency of *re-use* would appear relatively low, even though *reuse* is semantically compositional.

Lin et al. (2010) also provide a high-coverage, 10-million-phrase set of clusters extracted from the N-grams; we use these for our similarity features (Section 3.1.3). There are 1000 clusters in total. The data does not provide the context vectors for each phrase; rather, each phrase is listed with its 20 most similar clusters, measured by cosine similarity with the cluster centroid. We use these centroid similarities as values in a 1000-dimensional cluster-membership feature space. To calculate the similarity between two verbs, we calculate the cosine similarity between their cluster-membership vectors.

The feature classes in the following four subsections each make use of web-scale N-gram data.

3.1.1 HYPH features

Hyphenated verbs are usually compositional (e.g., *re-elect*). Of course, a particular instance of a compositional verb may or may not occur in hyphenated form. However, across a large corpus, compositional prefix verbs tend to occur in a hyphenated form more often than do non-compositional prefix verbs. We therefore provide real-valued features for how often the verb was hyphenated and unhyphenated on the web. For example, we collect counts for the frequencies of *re-elect* (33K) and *reelect* (9K) in our web corpus, and we convert the frequencies to log-counts. We also give real-valued features for the hyphenated/unhyphenated log-counts using only those occurrences of the verb that were *tagged* as a verb, exploiting the tag distributions in our web corpus as described above.

Nakov and Hearst (2005) previously used hyphenation counts as an indication of a syntactic relationship between nouns. In contrast, we leverage hyphenation counts as an indication of a semantic property of verbs.

3.1.2 COOC features

COOC features, and also the SIM (Section 3.1.3) and YAH (Section 3.2.2) features, concern the association in text between the prefix verb and its stem, where the stem occurs as a separate word. We call this the separated stem.

If a prefix verb is compositional, it is more likely to occur near its separated stem in text. We often see *agree* and *disagree*, *read* and *reread*, etc. occurring in the same segment of discourse. We create features for the association of the prefix verb and its separated stem in a discourse. We include the log-count of how often the verb and stem occur in the same N-gram (of length 2-to-5) in our N-gram corpus. Note that the 2-to-4-gram counts are not strictly a subset of the 5-gram counts, since fewer 5-grams pass the data's minimum frequency threshold.

We also include a real-valued pointwise mutual information (PMI) feature for the verb and separated stem's co-occurrence in an N-gram. For the PMI, we regard occurrence in an N-gram as an event, and calculate the probability that a verb and separated stem jointly occur in an N-gram, divided by the probability of their occurring in an N-gram independently.

3.1.3 SIM features

If a prefix verb is compositional, it should occur in similar contexts to its stem. The idea that a stem and stem+affix should be semantically similar has been exploited previously for morphological analysis (Schone and Jurafsky, 2000). We include a real-valued feature for the distributional similarity of the verb and stem using Lin's thesaurus (Lin, 1998). The coverage of this measure was low: it was non-zero for only 93 of the 1000 prefix verbs in our training set. We therefore also include distributional similarity calculated using the web-scale 10-million-phrase clustering as described above. Using this data, similarity is defined for 615 of the 1000 training verbs. We also explored a variety of WordNet-based similarity measures, but these ultimately did not prove helpful on development data.

3.1.4 FRQ features

We include real-valued features for the raw frequencies of the verb and the stem on the web. If these frequencies are widely different, it may indicate a non-compositional usage. Yarowsky and Wicentowski (2000) use similar statistics to identify words related by inflection, but they gather their counts from a much smaller corpus. In addition, higher-frequency prefix verbs may be *a priori* more likely to be non-compositional. A certain frequency is required for an irregular usage to become familiar to language speakers. The potential correlation between frequency and non-compositionality could thus also be exploited by the classifier via the FRQ features.

3.2 Other Features

3.2.1 LEX features

We provide lexical features for various aspects of a prefix verb. Binary features indicate the occurrence of particular verbs, prefixes, and stems, and whether the prefix verb is hyphenated. While hyphenated prefix verbs are usually compositional, even non-compositional prefix verbs may be hyphenated if the prefix and stem terminate and begin with a vowel, respectively. For example, non-compositional uses of *co-operate* are often hyphenated, whereas the compositional *remarry* is rarely hyphenated. We therefore have indicator features for the conjunction of the prefix and the first letter of the stem (e.g., *co-o*), and also for the prefix conjoined with a flag indicating whether the stem begins with a vowel (e.g., *co+vowel*).

3.2.2 YAH features

While the COOC features capture many cases where the verb and separated stem occur in close proximity (especially, but not limited to, conjunctions), there are many other cases where a longer distance might separate a compositional verb and its separated stem. For example, consider the sentence, "Brush the varnish on, but do not overbrush." Here, the verb and separated stem do not co-occur within a 5-gram window, and their co-occurrence will therefore not be recorded in our N-gram corpus. As an approximation for co-occurrence counts within a longer segment of discourse, we count the number of *pages* on the web where the verb and separated stem co-occur. We use hit-counts returned by the Yahoo search engine API.[1] Similar to our COOC features, we include a real-valued feature for the pointwise mutual information of the prefix verb and separated stem's co-occurrence on a web page, i.e., we use Turney's PMI-IR (Turney, 2001).

Baroni et al. (2002) use similar statistics to help discover morphologically-related words. In contrast to our features, however, their counts are derived from source text that is several orders of magnitude smaller in size.

[1] http://developer.yahoo.com/search/boss/

3.2.3 DIC features

One potentially useful resource, when available, is a dictionary of the conventional morphological segmentations of words in the language. Although these segmentations have been created for a different objective than that of our annotations, we hypothesize that knowledge of morphology can help inform our system's predictions. For each prefix verb, we include features for whether or not the prefix and stem are conventionally segmented into separate morphemes, according to a morphological dictionary. Similar to the count-based features, we include a DIC-undefined feature for the verbs that are not in the dictionary; any precompiled dictionary will have imperfect coverage of actual test examples.

Interestingly, DIC features are found to be among our least useful features in the final evaluation.

4 Experiments

4.1 Resources

We use CELEX (Baayen et al., 1996) as our dictionary for the DIC features. We also use CELEX to help extract our lists of prefixes and stems. We take every prefix that is marked in CELEX as forming a new verb by attaching to an existing verb. For stems, we use every verb that occurs in CELEX, but we also extend this list by automatically collecting a large number of words that were automatically tagged as verbs in the NYT section of Gigaword (Graff, 2003). To be included in the extra-verb list, a verb must occur more than ten times and be tagged as a verb more than 70% of the time by a part-of-speech tagger. We thereby obtain 43 prefixes and 6613 stems.[2] We aimed for an automatic, high-precision list for our initial experiments. This procedure is also amenable to human intervention; one could alternatively cast a wider net for possible stems and then manually filter false positives.

4.2 Annotated Data

We carried out a medium-scale annotation to provide training and evaluation data for our experiments.[3] The data for our annotations also comes from the NYT section of Gigaword. We first build a list of possible prefix verbs. We include any verb that a) is composed of a valid prefix and stem; and b) occurs at least twice in the corpus.[4] If the verb occurs less than 50 times in the corpus, we also require that it was tagged as a verb in at least 70% of cases. This results in 2077 possible prefix verbs for annotation.

For each verb type in our list of possible prefix verbs, we randomly select for annotation sentences from Gigaword containing the verb. We take at most three sentences for each verb type so that a few very common types (such as *become*, *understand*, and *improve*) do not comprise the majority of annotated examples. The resulting set of sentences includes a small number of sentences with incorrectly-tagged non-verbs; these are simply marked as non-verbs by our annotators and excluded from our final data sets. A graphical program was created for the annotation; the program automatically links to the online Merriam-Webster dictionary entries for the prefix verb and separated stem. When in doubt about a verb's meaning, our annotators adhere to the dictionary definitions. A single annotator labeled 1718 examples, indicating for each sentence whether the prefix verb was compositional. A second annotator then labeled a random subset of 150 of these examples, and agreement was calculated. The annotators agreed on 137 of the 150 examples. The *Kappa* statistic (Jurafsky and Martin, 2000, page 315), with P(E) computed from the confusion matrices, is 0.82, above the 0.80 level considered to indicate good reliability.

For our experiments, the 1718 annotated examples are randomly divided into 1000 training, 359 development, and 359 held-out test examples.

4.3 Classifier Settings

We train a linear support vector machine classifier using the efficient LIBLINEAR package (Fan et al., 2008). We use L2-loss and L2-regularization. We

[2] The 43 prefixes are: a- ab- ac- ad- as- be- circum- co- col- com- con- cor- counter- cross- de- dis- e- em- en- ex- fore- im- in- inter- ir- mis- out- over- per- photo- post- pre- pro- psycho- re- sub- super- sur- tele- trans- un- under- with-

[3] Our annotated data is publicly available at: http://www.cs.ualberta.ca/~ab31/verbcomp/

[4] We found that the majority of single-occurrence verbs in the Gigaword data were typos. We would expect true hapax legomena to be largely compositional, and we could potentially derive better statistics if we include them (Baayen and Sproat, 1996). One possible option, employed in previous work, is to ensure words of interest are "manually corrected for typing errors before further analysis" (Baayen and Renouf, 1996).

optimize the choice of features and regularization hyperparameter on development data, attaining a maximum when $C = 0.1$.

4.4 Evaluation

We compare the following systems:

1. **Base1**: always choose compositional (the majority class).
2. **Base2**: for each prefix, choose the majority class over the verbs having that prefix in training data.
3. **Morf**: the unsupervised Morfessor system (Creutz and Lagus, 2007) (Categories-ML, from 110K-word corpus). If Morfessor splits the prefix and stem into separate morphemes, we take the prediction as compositional. If it does anything else, we take it as non-compositional.
4. **SCD**: **S**upervised **C**ompositionality **D**etection: the system proposed in this paper.

We evaluate using *accuracy*: the percentage of examples classified correctly in held-out test data.

5 Results

We first analyze our annotations, gaining insight into the relation between our definition and conventional segmentations. We also note the consistency of our annotations across contexts. We then provide the main results of our system. Finally, we provide the results of our system when trained and tested on conventional morphological segmentations.

5.1 Analysis of Annotations

5.1.1 Annotation consistency with dictionaries

The majority of our examples are not present in a morphological dictionary, even in one as comprehensive as CELEX. The prefix verbs are in CELEX for only 670 of the 1718 total annotated instances.

For those that are in CELEX, Table 1 provides the confusion matrix that relates the CELEX segmentations to our annotations. The table shows that the major difference between our annotations and CELEX is that our definition of compositionality is more strict than conventional morphological segmentations. When CELEX does not segment the prefix from the stem (case 0), our annotations agree in 183 of 193 cases. When CELEX does split the prefix from the stem (case 1), the meaning is semantically compositional in less than half the cases. This is a key difference between conventional morphology and our semantic definition.

		CELEX segmentation	
		1	0
Compositionality	1	227	10
annotation	0	250	183

Table 1: Confusion matrix on the subset of prefix verb annotations that are also in CELEX. **1** indicates that the prefix and stem are segmented into separate morphemes, **0** indicates otherwise.

It is also instructive to analyze the 10 cases that are semantically compositional but which CELEX did not segment. Most of these are verbs that are conventionally viewed as single morphemes because they entered English as complete words. For example, *await* comes from the Old North French *awaitier*, itself from *waitier*. In practice, it is useful to know that *await* is compositional, i.e. that it can be rephrased as *wait for*. Downstream applications can exploit the compositionality of *await*, but miss the opportunity if using the conventional lack of segmentation.

5.1.2 Annotation consistency across contexts

We next analyze our annotated data to determine the consistency of compositionality across different occurrences of the same prefix-verb type. There are 1248 unique prefix verbs in our 1718 labeled examples: 45 verbs occur three times, 380 occur twice and 823 occur only once. Of the 425 verbs that occur multiple times, only 6 had different annotations in different examples (i.e., six verbs occur in both compositional and non-compositional usages in our dataset). These six instances are subtle, debatable, and largely uninteresting, depending on distinctions like whether the *proclaim* sense of *blazon* can substitute for the *celebrate* sense of *emblazon*, etc.

It is easy to find clearer ambiguities online, such as compositional examples of typically non-compositional verbs (how to *recover* a couch, when to *redress* a wound, etc.). However, in our data verbs like *recover* and *redress* always occur in their more dominant non-compositional sense. People may

Set	#	Base1	Base2	Morf	SCD
Test	359	65.7	87.2	73.8	93.6
∈ CELEX	128	30.5	73.4	50.8	89.8
∉ CELEX	231	85.3	94.8	86.6	95.7
∈ train	107	69.2	93.5	74.8	97.2
∉ train	252	64.3	84.5	73.4	92.1

Table 2: Number of examples (#) and accuracy (%) on test data, and on in-CELEX vs. not-in-CELEX, and in-training-data vs. not-in-training splits.

consciously or unconsciously recognize the possibility for confusion and systematically hyphenate prefixes from the stem if a less-common compositional usage is employed. For example, our data has "*repress* your feelings" for the non-compositional case but the hyphenated "*re-press* the center" for the compositional usage.[5]

Due to the consistency of compositionality across contexts, context-based *features* may simply not be very useful for classification. All the features we describe in Section 3 depend only on the prefix verb itself and not the verb context. Various context-dependent features did not improve accuracy on our development data and were thus excluded from the final system.

5.2 Main Results

The first row of Table 2 gives the results of all systems on test data. SCD achieves 93.6% accuracy, making one fifth as many errors as the majority-class baseline (Base1) and half as many errors as the more competitive prefix-based predictor (Base2). The substantial difference between SCD and Base2 shows that SCD is exploiting much information beyond the trivial memorization of a decision for each prefix. Morfessor performs better than Base1 but significantly worse than Base2. This indicates that state-of-the-art *unsupervised* morphological segmentation is not yet practical for semantic preprocessing. Of course, Morfessor was also designed with a different objective; in Section 5.3 we compare Morfessor and SCD on conventional morphological segmentations.

Prefix	# Tot	# Comp	SCD
re-	166	147	95.8
over-	26	25	96.2
out-	23	18	91.3
de-	21	**0**	100.0
pre-	19	16	94.7
un-	17	**1**	94.1
dis-	10	**0**	90.0
under-	9	7	77.8
co-	7	6	100.0
en-	5	2	60.0

Table 3: Total number of examples (# Tot), number of examples that are compositional (# Comp), and accuracy (%) of SCD on test data, by prefix.

We further analyzed the systems by splitting the test data two ways.

First, we separate verbs that occur in our morphological dictionary (∈ CELEX) from those that do not (∉ CELEX). Despite using the dictionary segmentation itself as a feature, the performance of SCD is worse on the ∈ CELEX verbs (89.8%). The comparison systems drop even more dramatically on this subset. The ∈ CELEX verbs comprise the more frequent, irregular verbs in English. Non-compositionality is the majority class on the examples that are in the dictionary.

On the other hand, one would expect verbs that are *not* in a comprehensive dictionary to be largely *compositional*, and indeed most of the ∉ CELEX verbs are compositional. However, there is still much to be gained from applying SCD, which makes a third as many errors as the system which always assigns compositional (95.7% for SCD vs. 85.3% for Base1).

Our second way of splitting the data is to divide our test set into prefix verbs that also occurred in training sentences (∈ train) and those that did not (∉ train). Over 70% did not occur in training. SCD scores 97.2% accuracy on those that did. The classifier is thus able to exploit the consistency of annotations across different contexts (Section 5.1.2). The 92.1% accuracy on the ∉-train portion also shows the features allow the system to generalize well to new, previously-unseen verbs.

Table 3 gives the results of our system on sets of

[5] Note that many examples like *recover*, *repress* and *redress* are only ambiguous in text, not in speech. Pronunciation reduces ambiguity in the same way that hyphens do in text. Conversely, observe that knowledge of compositionality could potentially help speech synthesis.

-LEX	-HYPH	-COOC	-SIM	-YAH	-FRQ	-DIC
85.0	92.8	92.5	93.6	93.6	93.6	93.6
85.5	93.6	92.8	93.0	93.3	93.9	
86.9	90.5	93.3	93.6	93.6		
84.1	90.3	93.3	93.6			
87.5	90.5	93.0				
85.5	89.4					

Table 4: Accuracy (%) of SCD as different feature classes are removed. Performance with all features is 93.6%.

verbs divided according to their prefix. The table includes those prefixes that occurred at least 5 times in the test set. Note that the prefixes have a long tail: these ten prefixes cover only 303 of the 359 test examples. Accuracy is fairly high across all the different prefixes. Note also that the three prefixes *de*-, *un*-, and *dis*- almost always correspond to non-compositional verbs. Each of these prefixes corresponds to a subtle form of negation, and it is usually difficult to paraphrase the negation using the stem. For example, *to demilitarize* does not mean *to not militarize* (or any other simple re-phrasing using the stem as a verb), and so our annotation marks it as non-compositional. Whether such a strict strategy is ultimately best may depend on the target application.

Feature Analysis

We perform experiments to evaluate which features are most useful for this task. Table 4 gives the accuracy of our system as different feature classes are *removed*. A similar table was previously used for feature analysis in Daumé III and Marcu (2005). Each row corresponds to performance with a group of features; each entry is performance with a particular feature class individually removed the group. We remove the least helpful feature class from each group in succession moving group-to-group down the rows.

We first remove the DIC features. These do not impact performance on test data. The last row gives the performance with only HYPH features (85.5, removing LEX), and only LEX features (89.4, removing HYPH). These are found to be the two most effective features for this task, followed by the COOC statistics. The other features, while marginally helpful on development data, are relatively ineffective on the test set. In all cases, removing LEX features hurts the most. Removing LEX not only removes useful stem, prefix, and hyphen information, but it also impairs the ability of the classifier to use the other features to separate the examples.

Base1	Base2	Morf	SCD
76.0	79.6	72.4	86.4

Table 5: Accuracy (%) on CELEX.

5.3 CELEX Experiments and Results

Finally, we train and test our system on prefix verbs where the segmentation decisions are provided by a morphological dictionary. We are interested in whether the strong results of our system could transfer to conventional morphological segmentations. We extract all verbs in CELEX that are valid verbs for our system (divisible into a prefix and verb stem), and take the CELEX segmentation as the label; i.e., whether the prefix and stem are separated into distinct morphemes. We extract 1006 total verbs.

We take 506 verbs for training, 250 verbs as a development set (to tune our classifier's regularization parameter) and 250 verbs as a final held-out test set. We use the same features and classifier as in our main results, except we remove the DIC features which are now the instance labels.

Table 5 shows the performance of our two baseline systems along with Morfessor and SCD. While the majority-class baseline is much higher, the prefix-based baseline is 7% *lower*, indicating that knowledge of prefixes, and lexical features in general, are less helpful for conventional segmentations. In fact, performance only drops 2% when we remove the LEX features, showing that web-scale information alone can enable solid performance on this task. Surprisingly, Morfessor performs worse here, below both baselines and substantially below the supervised system. We confirmed our Morfessor program was generating the same segmentations as the online demo. We also experimented with Linguistica (Goldsmith, 2001), training on a large corpus, but results were worse than with Morfessor.

Accurate segmentation of prefix verbs is clearly part of the mandate of these systems; prefix verb segmentation is simply a very challenging task. Unlike other, less-ambiguous tasks in morphology, a prefix/stem segmentation is plausible for all of our input verbs, since the putative morphemes are by definition valid morphemes in the language.

Overall, the results confirm and extend previous studies that show semantic information is helpful in morphology (Schone and Jurafsky, 2000; Yarowsky and Wicentowski, 2000). However, we reiterate that optimizing systems according to conventional morphology may not be optimal for downstream applications. Furthermore, accuracy is substantially lower in this setting than in our main results. Targeting conventional segmentations may be both more challenging and less useful than focusing on semantic compositionality.

6 Related Work

There is a large body of work on morphological analysis of English, but most of this work does not handle prefixes. Porter's stemmer is a well-known *suffix*-stripping algorithm (Porter, 1980), while publicly-available lemmatizers like *morpha* (Minnen et al., 2001) and PC-KIMMO (Karp et al., 1992) only process inflectional morphology. FreeLing (Atserias et al., 2006) comes with a few simple rules for deterministically stripping prefixes in some languages, but not English (e.g., only *semi*- and *re*- can be stripped when analyzing OOV Spanish verbs).

A number of modern morphological analyzers use supervised machine learning. These systems could all potentially benefit from the novel distributional features used in our model. Van den Bosch and Daelemans (1999) use memory-based learning to analyze Dutch. Wicentowski (2004)'s supervised WordFrame model includes a prefixation component. Results are presented on over 30 languages. Erjavec and Džeroski (2004) present a supervised lemmatizer for Slovene. Dreyer et al. (2008) perform supervised lemmatization on Basque, English, Irish and Tagalog; like us they include results when the set of lemmas is given. Toutanova and Cherry (2009) present a discriminative lemmatizer for English, Bulgarian, Czech and Slovene, but only handle suffix morphology. Poon et al. (2009) present an unsupervised segmenter, but one that is based on a log-linear model that can include arbitrary and interdependent features of the type proposed in our work. We see potential in combining the best elements of both approaches to obtain a system that does not need annotated training data, but can make use of powerful web-scale features.

Our approach follows previous systems for morphological analysis that leverage semantic as well as orthographic information (Yarowsky and Wicentowski, 2000; Schone and Jurafsky, 2001; Baroni et al., 2002). Similar problems also arise in core semantics, such as how to detect the compositionality of multi-word expressions (Lin, 1999; Baldwin et al., 2003; Fazly et al., 2009). Our problem is similar to the analysis of verb-particle constructions or VPCs (e.g., *round up, sell off*, etc.) (Bannard et al., 2003). Web-scale data can be used for a variety of problems in semantics (Lin et al., 2010), including classifying VPCs (Kummerfeld and Curran, 2008).

We motivated our work by describing applications in information retrieval, and here Google is clearly the elephant in the room. It is widely reported that Google has been using stemming since 2003; for example, a search today for *Porter stemming* returns pages describing the *Porter stemmer*, and the returned snippets have words like **stemming**, **stemmer**, and **stem** in bold text. Google can of course develop high-quality lists of morphological variants by paying attention to how users reformulate their queries. User query sessions have previously been used to expand queries using similar terms, such as substituting *feline* for *cat* (Jones et al., 2006). We show that high-quality, IR-friendly stemming is possible even without query data. Furthermore, query data could be combined with our other features for highly discriminative word stemming in context.

Beyond information retrieval, suffix-based stemming and lemmatization have been used in a range of NLP applications, including text categorization, textual entailment, and statistical machine translation. We believe accurate prefix-stripping can also have an impact in these areas.

7 Conclusions and Future Work

We presented a system for predicting the semantic compositionality of prefix verbs. We proposed a new, well-defined and practical definition of compositionality, and we annotated a corpus of sentences according to this definition. We trained a discriminative model to predict compositionality using a range of lexical and web-scale statistical features. Novel features include measures of the frequency of prefix-stem hyphenation, and statistics for the likelihood of the verb and stem co-occurring as separate words in an N-gram. The classifier is highly accurate across a range of prefixes, correctly predicting compositionality for 93.6% of examples.

Our preliminary results provide strong motivation for investigating and applying new distributional features in the prediction of both conventional morphology and in task-directed semantic compositionality. Our techniques could be used on a variety of other complex word forms. In particular, many of our features extend naturally to identifying stem-stem compounds (like *panfry* or *healthcare*). Also, it would be possible for our system to handle inflected forms by first converting them to their lemmas using a morphological analyzer. We could also jointly learn the compositionality of words across their inflections, along the lines of Yarowsky and Wicentowski (2000).

There are also other N-gram-derived features that warrant further investigation. One source of information that has not previously been exploited is the "lexical fixedness" (Fazly et al., 2009) of non-compositional prefix verbs. If prefix verbs are rarely rephrased in another form, they are likely to be non-compositional. For example, in our N-gram data, the count of *quest again* is relatively low compared to the count of *request*, indicating *request* is non-compositional. On the other hand, *marry again* is relatively frequent, indicating that *remarry* is compositional. Incorporation of these and other N-gram counts could further improve classification accuracy.

References

Jordi Atserias, Bernardino Casas, Elisabet Comelles, Meritxell González, Lluís Padró, and Muntsa Padró. 2006. FreeLing 1.3: Syntactic and semantic services in an open-source NLP library. In *LREC*.

R. Harald Baayen and Antoinette Renouf. 1996. Chronicling the Times: Productive lexical innovations in an English newspaper. *Language*, 72(1).

Harald Baayen and Richard Sproat. 1996. Estimating lexical priors for low-frequency morphologically ambiguous forms. *Comput. Linguist.*, 22(2):155–166.

R. Harald Baayen, Richard Piepenbrock, and Leon Gulikers. 1996. The CELEX2 lexical database. LDC96L14.

Timothy Baldwin, Colin Bannard, Takaaki Tanaka, and Dominic Widdows. 2003. An empirical model of multiword expression decomposability. In *ACL 2003 Workshop on Multiword Expressions*.

Colin Bannard, Timothy Baldwin, and Alex Lascarides. 2003. A statistical approach to the semantics of verb-particles. In *ACL 2003 Workshop on Multiword Expressions*.

Marco Baroni, Johannes Matiasek, and Harald Trost. 2002. Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In *ACL-02 Workshop on Morphological and Phonological Learning (SIGPHON)*, pages 48–57.

Matthew W. Bilotti, Boris Katz, and Jimmy Lin. 2004. What works better for question answering: Stemming or morphological query expansion? In *Information Retrieval for Question Answering (IR4QA) Workshop at SIGIR 2004*.

Thorsten Brants and Alex Franz. 2006. The Google Web 1T 5-gram Corpus Version 1.1. LDC2006T13.

Mathias Creutz and Krista Lagus. 2005. Inducing the morphological lexicon of a natural language from unannotated text. In *International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning*.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Trans. Speech Lang. Process.*, 4(1):1–34.

Hal Daumé III and Daniel Marcu. 2005. A large-scale exploration of effective global features for a joint entity detection and tracking model. In *HLT-EMNLP*.

Carl de Marken. 1996. Linguistic structure as composition and perturbation. In *ACL*.

Markus Dreyer, Jason Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *EMNLP*.

Tomaž Erjavec and Sašo Džeroski. 2004. Machine learning of morphosyntactic structure: Lemmatising unknown Slovene words. *Applied Artificial Intelligence*, 18:17–41.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *JMLR*, 9:1871–1874.

Afsaneh Fazly, Paul Cook, and Suzanne Stevenson. 2009. Unsupervised type and token identification of idiomatic expressions. *Comput. Linguist.*, 35(1):61–103.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Comput. Linguist.*, 27(2):153–198.

David Graff. 2003. English Gigaword. LDC2003T05.

Vera Hollink, Jaap Kamps, Christof Monz, and Maarten de Rijke. 2004. Monolingual document retrieval for European languages. *IR*, 7(1):33–52.

Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substitutions. In *WWW*.

Daniel Jurafsky and James H. Martin. 2000. *Speech and language processing*. Prentice Hall.

Daniel Karp, Yves Schabes, Martin Zaidel, and Dania Egedi. 1992. A freely available wide coverage morphological analyzer for English. In *COLING*.

Francis Katamba. 1993. *Morphology*. MacMillan Press.

Samarth Keshava and Emily Pitler. 2006. A simpler, intuitive approach to morpheme induction. In *2nd Pascal Challenges Workshop*.

Jonathan K. Kummerfeld and James R. Curran. 2008. Classification of verb particle constructions with the Google Web1T Corpus. In *Australasian Language Technology Association Workshop*.

Dekang Lin, Kenneth Church, Heng Ji, Satoshi Sekine, David Yarowsky, Shane Bergsma, Kailash Patil, Emily Pitler, Rachel Lathbury, Vikram Rao, Kapil Dalwani, and Sushant Narsale. 2010. New tools for web-scale N-grams. In *LREC*.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *COLING-ACL*.

Dekang Lin. 1999. Automatic identification of non-compositional phrases. In *ACL*.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

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

Preslav Nakov and Marti Hearst. 2005. Search engine statistics beyond the n-gram: Application to noun compound bracketing. In *CoNLL*.

Preslav Ivanov Nakov. 2007. *Using the Web as an Implicit Training Set: Application to Noun Compound Syntax and Semantics*. Ph.D. thesis, University of California, Berkeley.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *HLT-NAACL*.

Martin F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3).

Patrick Schone and Daniel Jurafsky. 2000. Knowledge-free induction of morphology using latent semantic analysis. In *LLL/CoNLL*.

Patrick Schone and Daniel Jurafsky. 2001. Knowledge-free induction of inflectional morphologies. In *NAACL*.

Kristina Toutanova and Colin Cherry. 2009. A global model for joint lemmatization and part-of-speech prediction. In *ACL-IJCNLP*.

Peter D. Turney. 2001. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *European Conference on Machine Learning*.

Antal Van den Bosch and Walter Daelemans. 1999. Memory-based morphological analysis. In *ACL*.

Richard Wicentowski. 2004. Multilingual noise-robust supervised morphological analysis using the word-frame model. In *ACL SIGPHON*.

Ying Xu, Christoph Ringlstetter, and Randy Goebel. 2009. A continuum-based approach for tightness analysis of Chinese semantic units. In *PACLIC*.

David Yarowsky and Richard Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *ACL*.

Joint Inference for Bilingual Semantic Role Labeling

Tao Zhuang and **Chengqing Zong**
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences
{tzhuang, cqzong}@nlpr.ia.ac.cn

Abstract

We show that jointly performing semantic role labeling (SRL) on bitext can improve SRL results on both sides. In our approach, we use monolingual SRL systems to produce argument candidates for predicates in bitext at first. Then, we simultaneously generate SRL results for two sides of bitext using our joint inference model. Our model prefers the bilingual SRL result that is not only reasonable on each side of bitext, but also has more consistent argument structures between two sides. To evaluate the consistency between two argument structures, we also formulate a log-linear model to compute the probability of aligning two arguments. We have experimented with our model on Chinese-English parallel PropBank data. Using our joint inference model, F1 scores of SRL results on Chinese and English text achieve 79.53% and 77.87% respectively, which are 1.52 and 1.74 points higher than the results of baseline monolingual SRL combination systems respectively.

1 Introduction

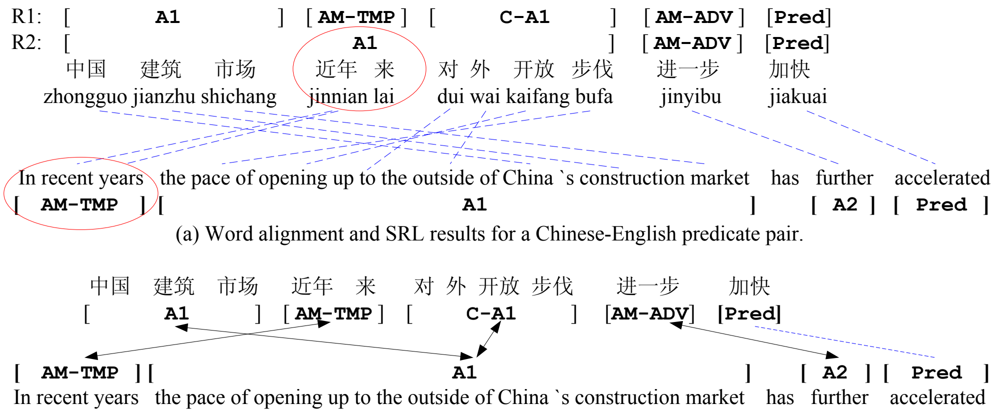
In recent years, there has been an increasing interest in SRL on several languages. However, little research has been done on how to effectively perform SRL on bitext, which has important applications including machine translation (Wu and Fung, 2009). A conventional way to perform SRL on bitext is performing SRL on each side of bitext separately, as has been done by Fung et al. (2007) on Chinese-English bitext. However, it is very difficult to obtain good SRL results on both sides of bitext in this way. The reason is that even the state-of-the-art SRL systems do not have very high accuracy on both English text (Màrquez et al., 2008; Pradhan et al., 2008; Punyakanok et al., 2008; Toutanova et al., 2008), and Chinese text (Che et al., 2008; Xue, 2008; Li et al., 2009; Sun et al., 2009).

On the other hand, the semantic equivalence between two sides of bitext means that they should have consistent predicate-argument structures. This bilingual argument structure consistency can guide us to find better SRL results. For example, in Figure 1(a), the argument structure consistency can guide us to choose a correct SRL result on Chinese side. Consistency between two argument structures is reflected by sound argument alignments between them, as shown in Figure 1(b). Previous research has shown that bilingual constraints can be very helpful for parsing (Burkett and Klein, 2008; Huang et al., 2008). In this paper, we show that the bilingual argument structure consistency can be leveraged to substantially improve SRL results on both sides of bitext.

Formally, we present a joint inference model to preform bilingual SRL. Using automatic word alignment on bitext, we first identify a pair of predicates that align with each other. And we use monolingual SRL systems to produce argument candidates for each predicate. Then, our model jointly generate SRL results for both predicates from their argument candidates, using integer linear programming (ILP) technique. An overview of our approach is shown in Figure 2.

Our joint inference model consists of three components: the source side, the target side, and the ar-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 304–314,
MIT, Massachusetts, USA, 9-11 October 2010.


R1: [A1] [AM-TMP] [C-A1] [AM-ADV] [Pred]
R2: [A1] [AM-ADV] [Pred]
中国 建筑 市场 近年 来 对 外 开放 步伐 进一步 加快
zhongguo jianzhu shichang jinnian lai dui wai kaifang bufa jinyibu jiakuai
In recent years the pace of opening up to the outside of China `s construction market has further accelerated
[AM-TMP] [A1] [A2] [Pred]
中国 建筑 市场 近年 来 对 外 开放 步伐 进一步 加快
[A1] [AM-TMP] [C-A1] [AM-ADV] [Pred]
[AM-TMP] [A1] [A2] [Pred]
In recent years the pace of opening up to the outside of China `s construction market has further accelerated

(a) Word alignment and SRL results for a Chinese-English predicate pair.

(b) Argument alignments for a Chinese-English predicate pair.

Figure 1: An example from Chinese-English parallel PropBank. In (a), the SRL results are generated by the state-of-the-art monolingual SRL systems. The English SRL result is correct. But it is to more difficult to get correct SRL result on Chinese side, because the AM-TMP argument embeds into a discontinuous A1 argument. The Chinese SRL result in the row marked by 'R1' is correct and consistent with the result on English side. Whereas the result in the row marked by 'R2' is incorrect and inconsistent with the result on English side, with the circles showing their inconsistency. The argument structure consistency can guide us to choose the correct Chinese SRL result.

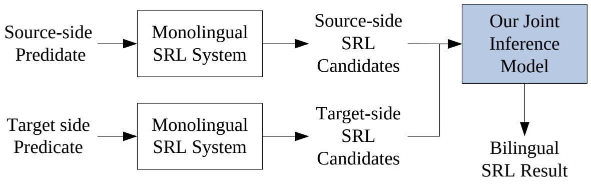


Figure 2: Overview of our approach.

gument alignment between two sides. These three components correspond to three interrelated factors: the quality of the SRL result on source side, the quality of the SRL result on target side, and the argument structure consistency between the SRL results on both sides. To evaluate the consistency between the two argument structures in our joint inference model, we formulate a log-linear model to compute the probability of aligning two arguments. Experiments on Chinese-English parallel PropBank shows that our model significantly outperforms monolingual SRL combination systems on both Chinese and English sides.

The rest of this paper is organized as follows: Section 2 introduces related work. Section 3 describes how we generate SRL candidates on each side of bitext. Section 4 presents our joint inference model. Section 5 presents our experiments. And Section 6 concludes our work.

2 Related Work

Some existing work on monolingual SRL combination is related to our work. Punyakanok et al. (2004; 2008) formulated an ILP model for SRL. Koomen et al. (2005) combined several SRL outputs using ILP method. Màrquez et al. (2005) and Pradhan et al. (2005) proposed combination strategies that are not based on ILP method. Surdeanu et al. (2007) did a complete research on a variety of combination strategies. Zhuang and Zong (2010) proposed a minimum error weighting combination strategy for Chinese SRL combination.

Research on SRL utilizing parallel corpus is also related to our work. Padó and Lapata (2009) did research on cross-lingual annotation projection on English-German parallel corpus. They performed SRL only on the English side, and then mapped the English SRL result to German side. Fung et al. (2007) did pioneering work on studying argument alignment on Chinese-English parallel PropBank. They performed SRL on Chinese and English sides separately. Then, given the SRL result on both sides, they automatically induced the argument alignment between two sides.

The major difference between our work and all existing research is that our model performs SRL inference on two sides of bitext simultaneously. In our model, we jointly consider three interrelated factors: SRL result on the source side, SRL result on the target side, and the argument alignment between them.

3 Generating Candidates for Inference

3.1 Monolingual SRL System

As shown in Figure 2, we need to use a monolingual SRL system to generate candidates for our joint inference model. We have implemented a monolingual SRL system which utilize full phrase-structure parse trees to perform SRL. In this system, the whole SRL process is comprised of three stages: pruning, argument identification, and argument classification. In the pruning stage, the heuristic pruning method in (Xue, 2008) is employed. In the argument identification stage, a number of argument locations are identified in a sentence. In the argument classification stage, each location identified in the previous stage is assigned a semantic role label. Maximum entropy classifier is employed for both the argument identification and classification tasks. And Zhang Le's MaxEnt toolkit[1] is used for implementation.

We use the monolingual SRL system described above for both Chinese and English SRL tasks. For the Chinese SRL task, the features used in this paper are the same with those used in (Xue, 2008). For the English SRL task, the features used are the same with those used in (Pradhan et al., 2008).

3.2 Output of the Monolingual SRL System

The maximum entropy classifier in our monolingual SRL system can output classification probabilities. We use the classification probability of the argument classification stage as an argument's probability. As illustrated in Figure 3, in an individual system's output, each argument has three attributes: its location in sentence loc, represented by the number of its first word and last word; its semantic role label l; and its probability p.

So each argument outputted by a system is a triple (loc, l, p). For example, the `A0` argument in Figure 3 is $((0, 2), \texttt{A0}, 0.94)$. Because these outputs are to be combined, we call such triple a **candidate**.

Sent:	外商 投资 企业	成为	中国 外贸 重要 增长点
Args:	[A0]	[Pred]	[A1]
loc:	(0, 2)		(4, 7)
l:	A0		A1
p:	0.94		0.92

Figure 3: Three attributes of an output argument: location loc, label l, and probability p.

3.3 Generating and Merging Candidates

To generate candidates for joint inference, we need to have multiple SRL results on each side of bitext. Therefore, for both Chinese and English SRL systems, we use the 3-best parse trees of Berkeley parser (Petrov and Klein, 2007) and 1-best parse trees of Bikel parser (Bikel, 2004) and Stanford parser (Klein and Manning, 2003) as inputs. All the three parsers are multilingual parsers. The second and third best parse trees of Berkeley parser are used for their good quality. Therefore, each monolingual SRL system produces 5 different outputs.

Candidates from different outputs may have the same loc and l but different p. So we merge all candidates with the same loc and l into one by averaging their probabilities. For a merged candidate (loc, l, p), we say that p is **the probability of assigning l to loc.**

4 Joint Inference Model

Our model can be conceptually decomposed to three components: the source side, the target side, and the argument alignment. The objective function of our joint inference model is the weighted sum of three sub-objectives:

$$\max\ O_s + \lambda_1 O_t + \lambda_2 O_a \tag{1}$$

where O_s and O_t represent the quality of the SRL results on source and target sides, and O_a represents the soundness of the argument alignment between the SRL results on two sides, λ_1, λ_2 are positive weights corresponding to the importance of O_t and O_a respectively.

4.1 Components of Source and Target Sides

4.1.1 Source Side Component

The source side component aims to improve the SRL result on source side. This is equivalent to a

[1] http://homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit .html

monolingual SRL combination problem.

For convenience, we denote the whole semantic role label set for source language as $\{l_1^s, l_2^s, \ldots, l_{L_s}^s\}$, in which $l_1^s \sim l_6^s$ stand for the key argument labels `A0` $\sim$ `A5` respectively. Suppose there are N_s different locations, denoted as $loc_1^s, \ldots, loc_{N_s}^s$, among all candidates on the source side. The probability of assigning l_j^s to loc_i^s is p_{ij}^s. An indicator variable x_{ij} is defined as:

$$x_{ij} = [loc_i^s \text{ is assigned label } l_j^s].$$

Then the source side sub-objective O_s in equation (1) is the sum of arguments' probabilities on source side:

$$O_s = \sum_{i=1}^{N_s} \sum_{j=1}^{L_s} (p_{ij}^s - T_s) x_{ij} \tag{2}$$

where T_s is a bias to prevent including too many candidates in solution (Surdeanu et al., 2007).

We consider the following two linguistically motivated constraints:

1. *No duplication*: There is no duplication for key arguments: `A0` $\sim$ `A5`.

2. *No overlapping*: Arguments cannot overlap with each other.

In (Punyakanok et al., 2004), several more constraints are considered. According to (Surdeanu et al., 2007), however, no significant performance improvement can be obtained by considering more constraints than the two above. So we do not consider other constraints.

The inequalities in (3) make sure that each loc_i^s is assigned at most one label.

$$\forall 1 \le i \le N_s : \sum_{j=1}^{L_s} x_{ij} \le 1 \tag{3}$$

The inequalities in (4) satisfy the *No duplication* constraint.

$$\forall 1 \le j \le 6 : \sum_{i=1}^{N_s} x_{ij} \le 1 \tag{4}$$

For any source side location loc_i^s, let C_i denote the index set of the locations that overlap with it. Then the *No overlapping* constraint means that if loc_i^s is assigned a label, i.e., $\sum_{j=1}^{N_s} x_{ij} = 1$, then for any $u \in C_i$, loc_u^s cannot be assigned any label, i.e., $\sum_{j=1}^{N_s} x_{uj} = 0$. A common technique in ILP modeling to form such a constraint is to use a sufficiently large auxiliary constant M. And the constraint is formulated as:

$$\forall 1 \le i \le N_s : \sum_{u \in C_i} \sum_{j=1}^{L_s} x_{uj} \le (1 - \sum_{j=1}^{L_s} x_{ij}) M \tag{5}$$

In this case, M only needs to be larger than the number of candidates to be combined. In this paper, $M = 500$ is large enough.

4.1.2 Target Side Component

In principle, the target side component of our joint inference model is the same with the source side component.

The whole semantic role label set for target language is denoted by $\{l_1^t, l_2^t, \ldots, l_{L_t}^t\}$. There are N_t different locations, denoted as $loc_1^t, \ldots, loc_{N_t}^t$, among all candidates in the target side. And $l_1^t \sim l_6^t$ stand for the key argument labels `A0` $\sim$ `A5` respectively. The probability of assigning l_j^t to loc_k^t is p_{kj}^t. An indicator variable y_{kj} is defined as:

$$y_{kj} = [loc_k^t \text{ is assigned label } l_j^t].$$

Then the target side sub-objective O_t in equation (1) is:

$$O_t = \sum_{k=1}^{N_t} \sum_{j=1}^{L_t} (p_{kj}^t - T_t) y_{kj} \tag{6}$$

The constraints on target side are as follows:

Each loc_k^t is assigned at most one label:

$$\forall 1 \le k \le N_t : \sum_{j=1}^{L_t} y_{kj} \le 1 \tag{7}$$

The *No duplication* constraint:

$$\forall 1 \le j \le 6 : \sum_{k=1}^{N_t} y_{kj} \le 1 \tag{8}$$

The *No overlapping* constraint:

$$\forall 1 \le k \le N_t : \sum_{v \in C_k} \sum_{j=1}^{L_t} y_{vj} \le (1 - \sum_{j=1}^{L_t} y_{kj}) M \tag{9}$$

In (9), C_k denotes the index set of the locations that overlap with loc_k^t, and the constant M is set to 500 in this paper.

4.2 Argument Alignment

The argument alignment component is the core of our joint inference model. It gives preference to the bilingual SRL results that have more consistent argument structures.

For a source side argument $arg_i^s = (loc_i^s, l^s)$ and a target side argument $arg_k^t = (loc_k^t, l^t)$, let z_{ik} be the following indicator variable:

$$z_{ik} = [arg_i^s \text{ aligns with } arg_k^t].$$

We use p_{ik}^a to represent the probability that arg_i^s and arg_k^t align with each other, i.e., $p_{ik}^a = P(z_{ik} = 1)$. We call p_{ik}^a the **argument alignment probability between** arg_i^s **and** arg_k^t.

4.2.1 Argument Alignment Probability Model

We use a log-linear model to compute the argument alignment probability p_{ik}^a between arg_i^s and arg_k^t. Let (s, t) denote a bilingual sentence pair and wa denote the word alignment on (s, t). Our log-linear model defines a distribution on z_{ik} given the tuple $tup = (arg_i^s, arg_k^t, wa, s, t)$:

$$P(z_{ik}|tup) \propto \exp(w^T \phi(tup))$$

where $\phi(tup)$ is the feature vector. With this model, p_{ik}^a can be computed as $p_{ik}^a = P(z_{ik} = 1|tup)$.

In order to study the argument alignment in corpus and to provide training data for our log-linear model, we have manually aligned the arguments in 60 files (chtb_0121.fid to chtb_0180.fid) of Chinese-English parallel PropBank. On this data set, we get the argument alignment matrix in Table 1.

Ch\En	A0	A1	A2	A3	A4	AM⋆	NUL
A0	492	30	4	0	0	0	46
A1	98	853	43	2	0	0	8
A2	9	57	51	1	0	47	0
A3	1	0	2	6	0	0	0
A4	0	0	2	0	3	0	0
AM⋆	0	2	39	0	0	895	221
NUL	53	14	27	0	0	45	0

Table 1: The argument alignment matrix on manually aligned corpus.

Each entry in Table 1 is the number of times for which one type of Chinese argument aligns with one type of English argument. AM⋆ stands for all adjuncts types like AM-TMP, AM-LOC, etc., and NUL means that the argument on the other side cannot be aligned with any argument on this side. For example, the number 46 in the A0 row and NUL column means that Chinese A0 argument cannot be aligned with any argument on English side for 46 times in our manually aligned corpus.

We use the following features in our model.

Word alignment feature: If there are many word-to-word alignments between arg_i^s and arg_k^t, then it is very probable that arg_i^s and arg_k^t would align with each other. We adopt the method used in (Padó and Lapata, 2009) to measure the word-to-word alignments between arg_i^s and arg_k^t. And the word alignment feature is defined as same as the *word alignment-based word overlap* in (Padó and Lapata, 2009). Note that this is a real-valued feature.

Head word alignment feature: The head word of an argument is usually more representative than other words. So we use whether the head words of arg_i^s and arg_k^t align with each other as a binary feature. The use of this feature is inspired by the work in (Burkett and Klein, 2008).

Semantic role labels of two arguments: From Table 1, we can see that semantic role labels of two arguments are a good indicator of whether they should align with each other. For example, a Chinese A0 argument aligns with an English A0 argument most of the times, and never aligns with an English AM⋆ argument in Table 1. Therefore, the semantic role labels of arg_i^s and arg_k^t are used as a feature.

Predicate verb pair: Different predicate pairs have different argument alignment patterns. Let's take the Chinese predicate *增长/zengzhang* and the English predicate *grow* as an example. The argument alignment matrix for all instances of the Chinese-English predicate pair (*zengzhang*, *grow*) in our manually aligned corpus is shown in Table 2.

CH \EN	A0	A1	A2	AM⋆	NUL
A0	0	16	0	0	0
A1	0	0	12	0	0
AM⋆	0	0	4	7	10
NUL	0	0	0	2	0

Table 2: The argument alignment matrix for the predicate pair (*zengzhang*, *grow*).

From Table 2 we can see that all A0 arguments of *zengzhang* align with A1 arguments of *grow*. This is very different from the results in Table 1, where a Chinese A0 argument tends to align with an English A0 argument. This phenomenon shows that a predicate pair can determine which types of arguments should align with each other. Therefore, we use the predicate pair as a feature.

4.2.2 Argument Alignment Component

The argument alignment sub-objective O_a in equation (1) is the sum of argument alignment probabilities:

$$O_a = \sum_{i=1}^{N_s} \sum_{k=1}^{N_t} (p_{ik}^a - T_a) z_{ik} \qquad (10)$$

where T_a is a bias to prevent including too many alignments in final solution, and p_{ik}^a is computed using the log-linear model described in subsection 4.2.1.

O_a reflects the consistency between argument structures on two sides of bitext. Larger O_a means better argument alignment between two sides, thus indicates more consistency between argument structures on two sides.

The following constraints are considered:

1. *Conformity with bilingual SRL result.* For all candidates on both source and target sides, only those that are chosen to be arguments on each side can be aligned.

2. *One-to-many alignment limit.* Each argument can not be aligned with more than 3 arguments.

3. *Complete argument alignment.* Each argument on source side must be aligned with at least one argument on target side, and vice versa.

The *Conformity with bilingual SRL result* constraint is necessary to validly integrate the bilingual SRL result with the argument alignment. This constraint means that if arg_i^s and arg_k^t align with each other, i.e., $z_{ik} = 1$, then loc_i^s must be assigned a label on source side, i.e., $\sum_{j=1}^{L_s} x_{ij} = 1$, and loc_k^t must be assigned a label on target side, i.e., $\sum_{j=1}^{L_t} y_{kj} = 1$. So this constraint can be represented as:

$$\forall 1 \leq i \leq N_s, 1 \leq k \leq N_t : \sum_{j=1}^{L_s} x_{ij} \geq z_{ik} \qquad (11)$$

$$\forall 1 \leq k \leq N_t, 1 \leq i \leq N_s : \sum_{j=1}^{L_t} y_{kj} \geq z_{ik} \qquad (12)$$

The *One-to-many alignment limit* constraint comes from our observation on manually aligned corpus. We have found that no argument aligns with more than 3 arguments in our manually aligned corpus. This constraint can be represented as:

$$\forall 1 \leq i \leq N_s : \sum_{k=1}^{N_t} z_{ik} \leq 3 \qquad (13)$$

$$\forall 1 \leq k \leq N_t : \sum_{i=1}^{N_s} z_{ik} \leq 3 \qquad (14)$$

The *Complete argument alignment* constraint comes from the semantic equivalence between two sides of bitext. For each source side location loc_i^s, if it is assigned a label, i.e., $\sum_{j=1}^{L_s} x_{ij} = 1$, then it must be aligned with some arguments on target side, i.e., $\sum_{k=1}^{N_t} z_{ik} \geq 1$. This can be represented as:

$$\forall 1 \leq i \leq N_s : \sum_{k=1}^{N_t} z_{ik} \geq \sum_{j=1}^{L_s} x_{ij} \qquad (15)$$

Similarly, each target side argument must be aligned to at least one source side argument. This can be represented as:

$$\forall 1 \leq k \leq N_t : \sum_{i=1}^{N_s} z_{ik} \geq \sum_{j=1}^{L_t} y_{kj} \qquad (16)$$

4.3 Complete Argument Alignment as a Soft Constraint

Although the hard *Complement argument alignment* constraint is ideally reasonable, in real situations this constraint does not always hold. The manual argument alignment result shown in Table 1 indicates that in some cases an argument cannot be aligned with any argument on the other side (see the NUL row and column in Table 1). Therefore, it would be reasonable to change the hard *Complement argument alignment* constraint to a soft one. To do so, we need to remove the hard *Complement argument alignment* constraint and add penalty for violation of this constraint.

If an argument does not align with any argument on the other side, we say it aligns with NUL. And we define the following indicator variables:

$z_{i,NUL} = [arg_i^s \text{ aligns with NUL}], 1 \leq i \leq N_s$.

$z_{NUL,k} = [arg_k^t$ aligns with $\mathtt{NUL}], 1 \leq k \leq N_t$. Then $\sum_{i=1}^{N_s} z_{i,NUL}$ is the number of source side arguments that align with NUL. And $\sum_{k=1}^{N_t} z_{NUL,k}$ is the number of target side arguments that align with NUL. For each argument that aligns with NUL, we add a penalty λ_3 to the argument alignment sub-objective O_a. Therefore, the sub-objective O_a in equation (10) is changed to:

$$O_a = \sum_{i=1}^{N_s} \sum_{k=1}^{N_t} (p_{ik}^a - T_a) z_{ik} \\ -\lambda_3 (\sum_{i=1}^{N_s} z_{i,NUL} + \sum_{k=1}^{N_t} z_{NUL,k}) \quad (17)$$

From the definition of $z_{i,NUL}$, it is obvious that, for any $1 \leq i \leq N_s$, $z_{i,NUL}$ and $z_{ik}(1 \leq k \leq N_t)$ have the following relationship: If $\sum_{k=1}^{N_t} z_{ik} \geq 1$, i.e., arg_i^s aligns with some arguments on target side, then $z_{i,NUL} = 0$; Otherwise, $z_{i,NUL} = 1$. These relationships can be captured by the following constraints:

$$\forall 1 \leq i \leq N_s, 1 \leq k \leq N_t : z_{i,NUL} \leq 1 - z_{ik} \quad (18)$$

$$\forall 1 \leq i \leq N_s : \sum_{k=1}^{N_t} z_{ik} + z_{i,NUL} \geq 1 \quad (19)$$

Similarly, for any $1 \leq k \leq N_t$, $z_{NUL,k}$ and $z_{ik}(1 \leq i \leq N_s)$ observe the following constraints:

$$\forall 1 \leq k \leq N_t, 1 \leq i \leq N_s : z_{NUL,k} \leq 1 - z_{ik} \quad (20)$$

$$\forall 1 \leq k \leq N_t : \sum_{i=1}^{N_s} z_{ik} + z_{NUL,k} \geq 1 \quad (21)$$

4.4 Models Summary

So far, we have presented two versions of our joint inference model. The first version treats *Complement argument alignment* as a hard constraint. We will refer to this version as *Joint1*. The objective function of *Joint1* is defined by equations (1, 2, 6, 10). And the constraints of *Joint1* are defined by equations (3-5, 7-9, 11-16).

The sencond version treats *Complement argument alignment* as a soft constraint. We will refer to this version as *Joint2*. The objective function of *Joint2* is defined by equations (1, 2, 6, 17). And the constraints of *Joint2* are defined by equations (3-5, 7-9, 11-14, 18-21).

Our baseline models are monolingual SRL combination models. We will refer to the source side combination model as *SrcCmb*. The objective of *SrcCmb* is to maximize O_s, which is defined in equation (2). And the constraints of *SrcCmb* are defined by equations (3-5). Similarly, we will refer to the target side combination model as *TrgCmb*. The objective of *TrgCmb* is to maximize O_t defined in equation (6). And the constraints of *TrgCmb* are defined by equations (7-9). In this paper, we employ lpsolve[2] to solve all ILP models.

5 Experiments

5.1 Experimental Setup

In our experiments, we use the Xinhua News portion of Chinese and English data in LDC OntoNotes Release 3.0. This data is a Chinese-English parallel proposition bank described in (Palmer et al., 2005). It contains parallel proposition annotations for 325 files (chtb_0001.fid to chtb_0325.fid) from Chinese-English parallel Treebank. The English part of this data contains proposition annotations only for verbal predicates. Therefore, we only consider verbal predicates in this paper.

We employ the GIZA++ toolkit (Och and Ney, 2003) to perform automatic word alignment. Besides the parallel PropBank data, we use additional 4,500K Chinese-English sentence pairs[3] to induce word alignments for both directions, with the default GIZA++ settings. The alignments are symmetrized using the intersection heuristic (Och and Ney, 2003), which is known to produce high-precision alignments.

We use 80 files (chtb_0001.fid to chtb_0080.fid) as test data, and 40 files (chtb_0081.fid to chtb_0120.fid) as development data. Although our joint inference model needs no training, we still need to train a log-linear argument alignment probability model, which is used in the joint inference model. As specified in subsection 4.2.1, the train-

[2] http://lpsolve.sourceforge.net/

[3] These data includes the following LDC corpus: LDC2002E18, LDC2003E07, LDC2003E14, LDC2005T06, LDC2004T07, LDC2000T50.

ing set for the argument alignment probability model consists of 60 files (chtb_0121.fid to chtb_0180.fid) with manual argument alignment. Unfortunately, the quality of automatic word alignment on one-to-many Chinese-English sentence pairs is usually very poor. So we only include one-to-one Chinese-English sentence pairs in all data. And not all predicates in a sentence pair can be included. Only bilingual predicate pairs are included. A bilingual predicate pair is defined to be a pair of predicates in bitext which align with each other in automatic word alignment. Table 3 shows how many sentences and predicates are included in each data set.

	Test	Dev	Train
Articles	1-80	81-120	121-180
Chinese Sentences	1067	578	778
English Sentences	1182	620	828
Bilingual pairs	821	448	614
Chinese Predicates	3792	2042	2572
English Predicates	2864	1647	1860
Bilingual pairs	1476	790	982

Table 3: Sentence and predicate counts.

Our monolingual SRL systems are trained separately. Our Chinese SRL system is trained on 640 files (chtb_0121.fid to chtb_0931.fid) in Chinese Propbank 1.0. Because Xinhua News is a quite different domain from WSJ, the training set for our English SRL system includes not only Sections 02∼21 of WSJ data in English Propbank, but also 205 files (chtb_0121.fid to chtb_0325.fid) in the English part of parallel PropBank. For Chinese, the syntactic parsers are trained on 640 files (chtb_0121.fid to chtb_0931.fid) plus the broadcast news portion of Chinese Treebank 6.0. For English, the syntactic parsers are trained on the following data: Sections 02∼21 of WSJ data in English Treebank, 205 files (chtb_0121.fid to chtb_0325.fid) of Xinhua News data in OntoNotes 3.0, and the Sinorama data in OntoNotes 3.0. We treat discontinuous and coreferential arguments in accordance to the CoNLL-2005 shared task (Carreras and Màrquez, 2005). The first part of a discontinuous argument is labeled as it is, and the second part is labeled with a prefix "C-". All coreferential arguments are labeled with a prefix "R-".

5.2 Tuning Parameters in Models

The models *Joint1*, *Joint2*, *SrcCmb*, and *TrgCmb* have different parameters. For each model, we have automatically tuned its parameters on development set using Powell's Mothod (Brent, 1973). Powell's Method is a heuristic optimization algorithm that does not require the objective function to have an explicit analytical formula. For a monolingual model like *SrcCmb* or *TrgCmb*, our objective is to maximize the F_1 score of the model's result on development set. But a joint model, like *Joint1* or *Joint2*, generates SRL results on both sides of bitext. So our objective is to maximize the sum of the two F_1 scores of the model's results for both Chinese and English on development set. For all models, we regard the parameters to be tuned as variables. Then we optimize our objective using Powell's Method. The solution of this optimization is the values of parameters. To avoid finding poor local optimum, we perform the optimization 30 times with different initial parameter values, and choose the best solution found. The final parameter values are listed in Table 4.

Model	T_s	T_t	T_a	λ_1	λ_2	λ_3
SrcCmb	0.21	-	-	-	-	-
TrgCmb	-	0.32	-	-	-	-
Joint1	0.17	0.22	0.36	0.96	1.04	-
Joint2	0.15	0.26	0.42	1.02	1.21	0.15

Table 4: Parameter values in models.

5.3 Individual SRL Outputs' Performance

As specified in subsection 3.3, the monoligual SRL system uses different parse trees to generate multiple SRL outputs. The performance of these outputs on test set is shown in Table 5. In Table 5, O1∼O3 are the outputs using 3-best parse trees of Berkeley parser respectively, O4 and O5 are the outputs using the best parse trees of Stanford parser and Bikel parser respectively.

As specified in subsection 5.1, only a small part of English SRL training data is in the same domain with test data. Therefore, the English SRL result in Table 5 is not very impressive. But the Chinese SRL result is pretty good.

Side	Outputs	P(%)	R(%)	F_1
Chinese	O1	**79.84**	**71.95**	**75.69**
	O2	78.53	70.32	74.20
	O3	78.41	69.99	73.96
	O4	73.21	67.13	70.04
	O5	75.32	63.78	69.07
English	O1	**77.13**	**70.42**	**73.62**
	O2	75.88	69.06	72.31
	O3	75.74	68.65	72.02
	O4	71.57	66.11	68.73
	O5	73.12	68.04	70.49

Table 5: The results of individual monolingual SRL outputs on test set.

5.4 Effects of Different Constraints

The *One-to-many limit* and *Complete argument alignment* constraints in subsection 4.2.2 comes from our empirical knowledge. To investigate the effect of these two constraits, we remove them from our joint inference models one by one, and observe the performance variations on test set. The results are shown in Table 6. In Table 6, 'c2' refers to the *One-to-many limit* constraint, 'c3' refers to the *Complete argument alignment* constraint, and '-' means removing. For example, '*Joint1* - c2' means removing the constraint 'c2' from the model *Joint1*. Recall that the only difference between *Joint1* and *Joint2* is that 'c3' is a hard constraint in *Joint1*, but a soft constraint in *Joint2*. Therefore, '*Joint2* - c3' and '*Joint2* - c2 - c3' do not appear in Table 6, because they are the same with '*Joint1* - c3' and '*Joint1* - c2 - c3' respectively.

Model	Side	P(%)	R(%)	F_1
Joint1	Chinese	82.95	75.21	78.89
Joint1 - c2		81.46	75.97	78.62
Joint1 - c3		82.36	74.68	78.33
Joint1 - c2 - c3		82.04	74.67	78.18
Joint2		**83.35**	**76.04**	**79.53**
Joint2 - c2		82.41	76.03	79.09
Joint1	English	79.38	75.16	77.21
Joint1 - c2		78.51	75.22	76.83
Joint1 - c3		78.66	74.55	76.55
Joint1 - c2 - c3		78.37	74.37	76.32
Joint2		**79.64**	**76.18**	**77.87**
Joint2 - c2		78.41	75.89	77.13

Table 6: Results of different joint models on test set.

From Table 6, we can see that the constraints 'c2' and 'c3' both have positive effect in our joint inference model, because removing any one of them causes performance degradation. And removing 'c3' from *Joint1* causes more performance degradation than removing 'c2'. This means that 'c3' plays a more important role than 'c2' in our joint inference model. Indeed, by treating 'c3' as a soft constraint, the model *Joint2* has the best performance on both sides of bitext.

5.5 Final Results

We use *Joint2* as our final joint inference model. And as specified in subsection 4.4, our baselines are monolingual SRL combination models: *SrcCmb* for Chinese, and *TrgCmb* for English. Note that *SrcCmb* and *TrgCmb* are basically the same as the state-of-the-art combination model in (Surdeanu et al., 2007) with *No overlapping* and *No duplication* constraints. The final results on test set are shown in Table 7.

Side	Model	P(%)	R(%)	F_1
Chinese	*SrcCmb*	82.58	73.92	78.01
	Joint2	**83.35**	**76.04**	**79.53**
English	*TrgCmb*	79.02	73.44	76.13
	Joint2	**79.64**	**76.18**	**77.87**

Table 7: Comparison between monolingual combination model and our joint inference model on test set.

From Table 5 and Table 7, we can see that *SrcCmb* and *TrgCmb* improve F_1 scores over the best individual SRL outputs by 2.32 points and 2.51 points on Chinese and English seperately. Thus they form strong baselines for our joint inference model. Even so, our joint inference model still improves F_1 score over *SrcCmb* by 1.52 points, and over *TrgCmb* by 1.74 points.

From Table 7, we can see that, despite only part of training data for English SRL system is in-domain, our joint inference model still produces good English SRL result. And the F_1 score of Chinese SRL result reaches 79.53%, which represents the state-of-the-art Chinese SRL performance to date.

6 Conclusions

In this paper, we propose a joint inference model to perform bilingual SRL. Our joint inference model incorporates not only linguistic constraints on source and target sides of bitext, but also the bilingual argument structure consistency requirement on bitext. Experiments on Chinese-English parallel PropBank show that our joint inference model is very effective for bilingual SRL. Compared to state-of-the-art monolingual SRL combination baselines, our joint inference model substantially improves SRL results on both sides of bitext. In fact, the solution of our joint inference model contains not only the SRL results on bitext, but also the optimal argument alignment between two sides of bitext. This makes our model especially suitable for application in machine translation, which needs to obtain the argument alignment.

Acknowledgments

The research work has been partially funded by the Natural Science Foundation of China under Grant No. 60975053 and 60736014, the National Key Technology R&D Program under Grant No. 2006BAH03B02. We would like to thank Jiajun Zhang for helpful discussions and the anonymous reviewers for their valuable comments.

References

Daniel Bikel. 2004. Intricacies of Collins Parsing Model. *Computational Linguistics*, 30(4):480-511.

Richard P. Brent. 1973. *Algorithms for Minimization without Derivatives*. Prentice-Hall, Englewood Cliffs, NJ.

David Burkett, and Dan Klein. 2008. Two Languages are Better than One (for Syntactic Parsing). In *Proceedings of EMNLP-2008*, pages 877-886.

Xavier Carreras, and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: semantic role labeling. In *Proceedings of CoNLL-2005*, pages 152-164.

Wanxiang Che, Min Zhang, Ai Ti Aw, Chew Lim Tan, Ting Liu, and Sheng Li. 2008. Using a Hybrid Convolution Tree Kernel for Semantic Role Labeling. *ACM Transactions on Asian Language Information Processing*, 2008, 7(4): 1-23.

Pascale Fung, Zhaojun Wu, Yongsheng Yang and Dekai Wu. 2007. Learning Bilingual Semantic Frames: Shallow Semantic Parsing vs. Semantic Role Projection. In *Proceedings of the 11th Conference on Theoretical and Methodological Issues in Machine Translation*, pages 75-84.

Liang Huang, Wenbin Jiang, Qun Liu. 2009. Bilingually-Constrained (Monolingual) Shift-Reduce Parsing. In *Proceedings of EMNLP-2009*, pages 1222-1231.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL-2003*, pages 423-430.

Peter Koomen, Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2005. Generalized Inference with Multiple Semantic Role Labeling Systems. In *Proceedings of CoNLL-2005 shared task*, pages 181-184.

Junhui Li, Guodong Zhou, Hai Zhao, Qiaoming Zhu, and Peide Qian. 2009. Improving Nominal SRL in Chinese Language with Verbal SRL Information and Automatic Predicate Recognition. In *Proceedings of EMNLP-2009*, pages 1280-1288.

Lluís Màrquez, Xavier Carreras, Kenneth C. Litkowski, Suzanne Stevenson. 2008. Semantic Role Labeling: An Introduction to the Special Issue. *Computational Linguistics*, 34(2):145-159.

Lluís Màrquez, Mihai Surdeanu, Pere Comas, and Jordi Turmo. 2005. A Robust Combination Strategy for Semantic Role Labeling. In *Proceedings of EMNLP-2005*, pages 644-651.

Frans J. Och, and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29:19-51.

Sebastian Padó, and Mirella Lapata. 2009. Cross-lingual Annotation Projection of Semantic Roles. *Journal of Artificial Intelligence Research (JAIR)*, 36:307-340.

Martha Palmer, Nianwen Xue, Olga Babko-Malaya, Jinying Chen, Benjamin Snyder. 2005. A Parallel Proposition Bank II for Chinese and English. In *Frontiers in Corpus Annotation, Workshop in conjunction with ACL-05*, pages 61-67.

Slav Petrov and Dan Klein. 2007. Improved Inference for Unlexicalized parsing. In *Proceedings of ACL-2007*, pages 46-54.

Sameer S. Pradhan, Wayne Ward, Kadri Hacioglu, James H. Martin, and Daniel Jurafsky. 2005. Semantic Role Labeling Using Different Syntactic Views. In *Proceedings of ACL-2005*, pages 581-588.

Sameer S. Pradhan, Wayne Ward, James H. Martin. 2008. Towards Robust Semantic Role Labeling. *Computational Linguistics*, 34(2):289-310.

Vasin Punyakanok, Dan Roth, Wen-tauYih. 2008. The Importance of Syntactic Parsing and Inference in Semantic Role Labeling. *Computational Linguistics*, 34(2):257-287.

Vasin Punyakanok, Dan Roth, Wen-tau Yih, and Dav Zimak. 2004. Semantic Role Labeling via Integer Linear Programming Inference. In *Proceedings of COLING-2004*, pages 1346-1352.

Weiwei Sun, Zhifang Sui, Meng Wang, and Xin Wang. 2009. Chinese Semantic Role Labeling with Shallow

Parsing. In *Proceedings of EMNLP-2009*, pages 1475-1483.

Mihai Surdeanu, Lluís Màrquez, Xavier Carreras, and Pere R. Comas. 2007. Combination Strategies for Semantic Role Labeling. *Journal of Artificial Intelligence Research (JAIR)*, 29:105-151.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A Global Joint Model for Semantic Role Labeling. *Computational Linguistics*, 34(2): 145-159.

Dekai Wu, and Pascale Fung. 2009. Semantic Roles for SMT: A Hybrid Two-Pass Model. In *Proceedings of NAACL-2009*, pages 13-16.

Nianwen Xue. 2008. Labeling Chinese Predicates with Semantic Roles. *Computational Linguistics*, 34(2): 225-255.

Tao Zhuang, and Chengqing Zong. 2010. A Minimum Error Weighting Combination Strategy for Chinese Semantic Role Labeling. In *Proceedings of COLING-2010*, pages 1362-1370.

Automatic Discovery of Manner Relations and its Applications

Eduardo Blanco and **Dan Moldovan**
Human Language Technology Research Institute
The University of Texas at Dallas
Richardson, TX 75080 USA
{eduardo,moldovan}@hlt.utdallas.edu

Abstract

This paper presents a method for the automatic discovery of MANNER relations from text. An extended definition of MANNER is proposed, including restrictions on the sorts of concepts that can be part of its domain and range. The connections with other relations and the lexico-syntactic patterns that encode MANNER are analyzed. A new feature set specialized on MANNER detection is depicted and justified. Experimental results show improvement over previous attempts to extract MANNER. Combinations of MANNER with other semantic relations are also discussed.

1 Introduction

Extracting semantic relations from text is an important step towards understanding the meaning of text. Many applications that use no semantics, or only shallow semantics, could benefit by having available more text semantics. Recently, there is a growing interest in text semantics (Màrquez et al., 2008; Davidov and Rappoport, 2008).

An important semantic relation for many applications is the MANNER relation. Broadly speaking, MANNER encodes the mode, style, way or fashion in which something is done or happened. For example, *quick delivery* encodes a MANNER relation, since *quick* is the manner in which the *delivery* happened.

An application of MANNER detection is Question Answering, where many *how* questions refer to this particular relation. Consider for example the question *How did the President communicate his message?*, and the text *Through his spokesman, Obama sent a strong message [. . .]*. To answer such questions, it is useful to identify first the MANNER relations in text.

MANNER occurs frequently in text and it is expressed by a wide variety of lexico-syntactic patterns. For example, PropBank annotates 8,037 ARGM-MNR relations (10.7%) out of 74,980 adjunct-like arguments (ARGMs). There are verbs that state a particular way of doing something, e.g., *to limp* implicitly states a particular way of *walking*. Adverbial phrases and prepositional phrases are the most productive patterns, e.g., *The nation's industrial sector is now growing very slowly if at all* and *He started the company on his own*. Consider the following example: *The company said Mr. Stronach will personally direct the restructuring assisted by Manfred Gingl, [. . .]*[1]. There are two MANNER relations in this sentence: the underlined chunks of text encode the way in which *Mr. Stronach will direct the restructuring*.

2 Previous Work

The extraction of semantic relations in general has caught the attention of several researchers. Approaches to detect semantic relations usually focus on particular lexical and syntactic patterns. There are both unsupervised (Davidov et al., 2007; Turney, 2006) and supervised approaches. The SemEval-2007 Task 04 (Girju et al., 2007) aimed at relations between nominals. Work has been done on detecting relations within noun phrases (Nulty, 2007),

[1] Penn TreeBank, file `wsj_0027`, sentence 10.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 315–324,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

named entities (Hirano et al., 2007), clauses (Szpakowicz et al., 1995) and syntax-based comma resolution (Srikumar et al., 2008). There have been proposals to detect a particular relation, e.g., CAUSE (Chang and Choi, 2006), INTENT (Tatu, 2005), PART-WHOLE (Girju et al., 2006) and IS-A (Hearst, 1992).

MANNER is a frequent relation, but besides theoretical studies there is not much work on detecting it. Girju et al. (2003) propose a set of features to extract MANNER exclusively from adverbial phrases and report a precision of 64.44% and recall of 68.67%. MANNER is a semantic role, and all the works on the extraction of roles (Gildea and Jurafsky, 2002; Giuglea and Moschitti, 2006) extracts MANNER as well. However, these approaches treat MANNER as any other role and do not use any specific features for its detection. As we show in this paper, MANNER has its own unique characteristics and identifying them improves the extraction accuracy. The two most used semantic role annotation resources, FrameNet (Baker et al., 1998) and PropBank (Palmer et al., 2005), include MANNER.

The main contributions of this paper are: (1) empirical study of MANNER and its semantics; (2) analysis of the differences with other relations; (3) lexico-syntactic patterns expressing MANNER; (4) a set of features specialized on the detection of MANNER; and (5) the way MANNER combines with other semantic relations.

3 The Semantics of MANNER Relation

Traditionally, a semantic relation is defined by stating the kind of connection linking two concepts. For example, MANNER is loosely defined by the PropBank annotation guidelines[2] as *manner adverbs specify how an action is performed [...] manner should be used when an adverb be an answer to a question starting with 'how?'*. We find this kind of definition weak and prone to confusion (Section 3.2). Nonetheless, to the best of our knowledge, semantic relations have been mostly defined stating only a vague definition.

Following (Helbig, 2005), we propose an extended definition for semantic relations, including semantic restrictions for its domain and range. These restrictions help deciding which relation holds between a given pair of concepts. A relation shall not hold between two concepts unless they belong to its domain and range. These restrictions are based on theoretical and empirical grounds.

3.1 MANNER Definition

Formally, MANNER is represented as MNR(x, y), and it should be read x *is the manner in which* y *happened*. In addition, DOMAIN(MNR) and RANGE(MNR) are the sets of sorts of concepts that can be part of the first and second argument.

RANGE(MNR), namely y, is restricted to situations, which are defined as *anything that happens at a time and place*. Situations include events and states and can be expressed by verbs or nouns, e.g., *conference*, *race*, *mix* and *grow*. DOMAIN(MNR), namely x, is restricted to qualities (ql), non temporal abstract objects (ntao) and states (st). Qualities represent characteristics that can be assigned to other concepts, such as *slowly* and *abruptly*. Non temporal abstract objects are intangible entities. They are somehow product of human reasoning and are not palpable. They do not encode periods or points of time, such as *week*, or *yesterday*. For example, *odor*, *disease*, and *mile* are ntao; *book* and *car* are not because they are tangible. Unlike events, states are situations that do not imply a change in the concepts involved. For example, *standing there* or *holding hands* are states; whereas *walking to the park* and *pinching him* are events. For more details about these semantic classes, refer to (Helbig, 2005).

These semantic restrictions on MANNER come after studying previous definitions and manual examination of hundreds of examples. Their use and benefits are described in Section 4.

3.2 MANNER and Other Semantic Relations

MANNER is close in meaning with several other relations, specifically INSTRUMENT, AT-LOCATION and AT-TIME.

Asking *how* does not identify MANNER in many cases. For example, given *John broke the window [with a hammer]*, the question *how did John break the window?* can be answered by *with the hammer*, and yet *the hammer* is not the MANNER but the INSTRUMENT of the *broke* event. Other relations that

[2] http://verbs.colorado.edu/˜mpalmer/projects/ace/PBguidelines.pdf, page 26.

may be confused as MANNER include AT-LOCATION and AT-TIME, like in *[The dog jumped]$_x$ [over the fence]$_y$* and *[John used to go]$_x$ [regularly]$_y$*.

A way of solving this ambiguity is by prioritizing the semantic relations among the possible candidates for a given pair of concepts. For example, if both INSTRUMENT and MANNER are possible, the former is extracted. In a similar fashion, AT-LOCATION and AT-TIME could have higher priority than MANNER. This idea has one big disadvantage: the correct detection of MANNER relies on the detection of several other relations, a problem which has proven difficult and thus would unnecessarily introduce errors.

Using the proposed extended definition one may discard the false MANNER relations above. *Hammer* is not a quality, non temporal abstract object or state (*hammers* are palpable objects), so by definition a relation of the form MNR(*the hammer*, *y*) shall not hold. Similarly, *fence* and *week* do not fulfill the domain restriction, so MNR(*over the fence*, *y*) and MNR(*every other week*, *y*) are not valid either.

MANNER also relates to CAUSE. Again, asking *how?* does not resolve the ambiguity. Given *The legislation itself noted that it [was introduced]$_x$ "by request," [...]* (wsj_0041, 47), we believe the underlined PP indicates the CAUSE and not the MANNER of *x* because the *introduction of the legislation* is the effect of the *request*. Using the extended definition, since *request* is an event (it implies a change), MNR(*by request*, *y*) is discarded based on the domain and range restrictions.

4 Argument Extraction

In order to implement domain and range restrictions, one needs to map words to the four proposed semantic classes: situations (si), states (st), qualities (ql) and non temporal abstract objects (ntao). These classes are the ones involved in MNR; work has been done to define in a similar way more relations, but we do not report on that in this paper.

First, the head word of a potential argument is identified. Then, the head is mapped into a semantic class using three sources of information: POS tags, WordNet hypernyms and named entity (NE) types. Table 1 presents the rules that define the mapping. We obtained them following a data-driven approach using a subset of MANNER annotation from PropBank and FrameNet. Intermediate classes are defined to facilitate legibility; intermediate classes ending in -NE only involve named entity types.

Words are automatically POS tagged using a modified Brill tagger. We do not perform word sense disambiguation because in our experiments it did not bring any improvement; all senses are considered for each word. isHypo(x) for a given word *w* indicates if any of the senses of *w* is a hyponym of *x* in WordNet 2.0. An in-house NE recognizer is used to assign NE types. It detects 90 types organized in a hierarchy with an accuracy of 92% and it has been used in a state-of-the-art Question Answering system (Moldovan et al., 2007). As far as the mapping is concerned, only the following NE types are used: *human*, *organization*, *country*, *town*, *province*, *other-loc*, *money*, *date* and *time*. The mapping also uses an automatically built list of verbs and nouns that encode events (verb_events and noun_events).

The procedure to map words into semantic classes has been evaluated on a subset of PropBank which was not used to define the mapping. First, we selected 1,091 sentences which contained a total of 171 MANNER relations. We syntactically parsed the sentences using Charniak's parser and then performed argument detection by matching the trees to the syntactic patterns depicted in Section 5. 52,612 arguments pairs were detected as potential MANNER. Because of parsing errors, 146 (85.4%) of the 171 MANNER relations are in this set.

After mapping and enforcing domain and range constraints, the argument pairs were reduced to 11,724 (22.3%). The filtered subset includes 140 (81.8%) of the 171 MANNER relations. The filtering does make mistakes, but the massive pruning mainly filters out potential relations that do not hold: it filters 77.7% of argument pairs and it only misclassifies 6 pairs.

5 Lexico-Syntactic Patterns Expressing MANNER

MANNER is expressed by a wide variety of lexico-syntactic patterns, implicitly or explicitly.

Table 2 shows the syntactic distribution of MANNER relation in PropBank. We only consider relations between a single node in the syntactic tree and

Class	Rule
situation	`state || event`
state	`POStag=verb || isHypo(state.n.4)`
event	`(POStag=verb && in(verb_events)) || (POStag=noun && !animate_object && (isHypo(phenomenon.n.1) || isHypo(event.n.1) || in(noun_events)))`
animate_object	`livingNE || (POStag=noun && ((isHypo(entity.n.1) && !isHypo(thing.n.9) && !isHypo(anticipation.n.4)) || isHypo(social_group.n.1)))`
livingNE	`neType=(human | organization | country | town | province | other-loc)`
quality	`POStag=(adverb | gerund) || headPP=(with | without)`
non_temporal_abstract_object	`abstract_object && !temporal`
abstract_object	`neType=money || isHypo(thing.n.9) || (!isHypo(social_group.n.1) && (isHypo(abstraction.n.6 | psychological_feature.n.1 | possession.n.2 | event.n.1 | state.n.4 | group.n.1 | act.n.2)))`
temporal	`TemporalNE || isHypo(time_period.n.1) || isHypo(time.n.5)`
temporalNE	`ne-type=(date | time)`

Table 1: Mapping for the semantic classes used for defining DOMAIN(MNR) and RANGE(MNR).

.

Synt. pattern	#Occ.	%Occ.	Example	
			File, #sent	Sentence
ADVP	3559	45.3%	wsj_0039, 24	This story line might [resonate]$_y$ [more strongly]$_{ADVP}$ if Mr. Lane had as strong a presence in front of the camera as he does behind it.
PP	3499	44.6%	wsj_2451, 0	NBC may yet find a way to [take]$_y$ a passive, minority interest in a program-maker [without violating the rules]$_{PP}$.
RB	286	3.6%	wsj_0052, 3	Backe is [a [[closely]$_{RB}$ [held]$_y$]$_{ADJP}$ media firm]$_{NP}$ run by former CBS Inc. President Jon Backe.
S	148	1.9%	wsj_1217, 25	Salomon [posted]$_y$ an unexpectedly big gain in quarterly earnings, [aided by its securities trading and investments banking activities]$_S$.
NP	120	1.5%	wsj_0100, 21	[...] he [graduated]$_y$ [Phi Beta Kappa]$_{NP}$ from the University of Kentucky at age 18, after spending only 2 1/2 years in college.
Other	240	3.1%	wsj_1337, 0	Tokyo stocks [closed]$_y$ [firmer]$_{ADJP}$ Monday, with the Nikkei index making its fifth consecutive daily gain.

Table 2: Syntactic patterns encoding MANNER in PropBank, number of occurrences, and examples. A total of 7,852 MANNER relations are encoded in PropBank between a single node in the syntactic tree and a verb. In all examples, MNR(x, y) holds, where x is the text underlined. Syntactic annotation comes straight from the Penn TreeBank.

a verb; MANNER relations expressed by trace chains identifying coreference and split arguments are ignored. This way, we consider 7,852 MANNER out of the total of the 8,037 PropBank annotates. Because ADVPs and PPs represent 90% of MANNER relations, in this paper we focus exclusively on these two phrases.

For both ADVP and PP the most common direct ancestor is either a VP or S, although examples are found that do not follow this rule. Table 3 shows the number of occurrences for several parent nodes and examples. Only taking into account phrases whose ancestor is either a VP or S yields a coverage of 98% and thus those are the focus of this work.

5.1 Ambiguities of MANNER

Both ADVPs and PPs are highly ambiguous when the task is to identify their semantics. The PropBank authors (Palmer et al., 2005) report discrepancies between annotators mainly with AT-LOCATION and simply no relation, i.e., when a phrase does not encode a role at all. In their annotation, 22.2% of ADVPs encode MANNER (30.3% AT-TIME), whereas only 4.6% of PPs starting with *in* and 6.1% start-

Parent	#Occ.	Example		
		Phrase	File, #sent	Sentence
VP	3306	ADVP	wsj_2341, 23	The company [was [officially]$_{ADVP}$ [merged]$_{y}$ with Bristol-Myers Co. earlier this month]$_{VP}$.
	3107	PP	wsj_2320, 7	This is something P&G [would [do]$_{y}$ [with or without Kao]$_{PP}$]$_{VP}$, says Mr. Zurkuhlen.
S	215	ADVP	wsj_0044, 6	[[Virtually word by word]$_{ADVP}$, the notes [matched]$_{y}$ questions and answers on the social-studies section of the test the student was taking.]$_{S}$
	339	PP	wsj_2454, 9	[[Under the laws of the land]$_{PP}$, the ANC [remains]$_{y}$ an illegal organization, and its headquarters are still in Lusaka, Zambia.]$_{S}$
ADJP	17	ADVP	wsj_1057, 85	[...] ABC touted "Call to Glory," but the military drama was [[missing]$_{y}$ [in action]$_{PP}$]$_{ADJP}$ within weeks.
	4	PP	wsj_2431, 14	Two former ministers [were]$_{y}$ [[so heavily]$_{ADVP}$ implicated]$_{ADJP}$ in the Koskotas affair that PASOK members of Parliament voted [...]
PP	9	ADVP	wsj_1249, 24	In Japan, by contrast, companies tend to develop their talent and [promote]$_{y}$ [from [within]$_{PP}$]$_{PP}$.
	9	PP	wsj_1505, 30	London share prices were [influenced]$_{y}$ [[largely]$_{ADVP}$ by declines on Wall Street and weakness in the British pound]$_{PP}$.

Table 3: Examples of ADVPs and PPs encoding MANNER with different nodes as parents. In all examples, MNR(x, y) holds, where x is the underlined phrase. Syntactic annotation comes straight from the Penn TreeBank.

ing with *at* encode MANNER. The vast majority of PPs encode either a AT-TIME or AT-LOCATION.

MANNER relations expressed by ADVPs are easier to detect since the adverb is a clear signal. Adverbs ending in *-ly* are more likely to encode a MANNER. Not surprisingly, the verb they attach to also plays an important role. Section 6.2 depicts the features used.

PPs are more complicated since the preposition per se is not a signal of whether or not the phrase encodes a MANNER. Even prepositions such as *under* and *over* can introduce a MANNER. For example, *A majority of an NIH-appointed panel recommended late last year that the research continue under carefully controlled conditions, [...]* (wsj_0047, 9) and *[...] bars where Japanese revelers sing over recorded music, [...]* (wsj_0300, 3). Note that in both cases, the head of the NP contained in the PP encoding MANNER (*conditions* and *music*) belongs to ntao (Section 4). Other prepositions, like *with* and *like* are more likely to encode a MANNER, but again it is not guaranteed.

6 Approach

We propose a supervised learning approach, where instances are positive and negative MANNER examples. Due to their intrinsic difference, we build different models for ADVPs and PPs.

6.1 Building the Corpus

The corpus building procedure is as follows. First, all ADVPs and PPs whose parent node is a VP or S and encode a MANNER according to PropBank are extracted, yielding 3559 and 3499 positive instances respectively. Then, 10,000 examples of ADVPs and another 10,000 of PPs from the Penn TreeBank not encoding a MANNER according to PropBank are added. These negative instances must have as their parent node either VP or S as well and are selected randomly.

The total number of instances, 13,559 for ADVPs and 13,499 for PPs, are then divided into training (60%), held-out (20%) and test (20%). The held-out portion is used to tune the feature set and the final results provided are the ones obtained with the test portion, i.e., instances that have not been used in any way to learn the models. Because PropBank adds semantic role annotation on top of the Penn TreeBank, we have gold syntactic annotation for all instances.

6.2 Selecting features

Selected features are derived from previous works on detecting semantic roles, namely (Gildea and Jurafsky, 2002) and the participating systems in

No.	Feature	Values	Explanation
1	parent-node	{S, VP}	syntactic node of ADVP's parent
2	num-leaves	ℕ	number of words in ADVP
3	**adverb**	{often, strongly, . . . }	main adverb of ADVP
4	**dictionary**	{yes, no}	is adverb is in dictionary?
5	**ends-with-ly**	{yes, no}	does adverb end in *-ly*?
6	POS-tag-bef	POStags	POS tag word before adverb
7	POS-tag-aft	POStags	POS tag word after adverb
8	verb	{assigned, go, . . . }	main verb the ADVP attaches to
9	distance	ℕ	number of words between adverb and verb

Table 4: Features used for extracting MANNER from ADVPs, their values and explanation. Features 4 and 5 are specialized on MANNER detection.

No.	Feature	Values	Explanation
1	parent-node	{S, VP}	syntactic node of PP's parent
2	next-node	{NP, SBAR, , . . . }	syntactic node of sibling to the right of PP
3	**num-pp-bef**	ℕ	number of sibling before PP which are PP
4	**num-pp-aft**	ℕ	number of sibling after PP which are PP
5	first-word	{with, after, . . . }	first word in PP
6	first-POS-tag	POStags	first POS tag in PP
7	**first-prep**	{by, on, . . . }	first preposition in PP
8	POS-tag-bef	POStags	POS tag before first-word
9	POS-tag-aft	POStags	POs tag after first-word
10	word-aft	{one, their, . . . }	word after first-word
11	**has-rb**	{yes, no}	does the PP contain an adverb?
12	**has-quotes**	{yes, no}	does the PP have any quotes?
13	**head-np-lemma**	{amount, year, . . . }	head of the NP whose parent is the PP
14	**head-is-last**	{yes, no}	is head-np the last word of the sentence?
15	**head-has-cap**	{yes, no}	does the PP have a capitalized word?
16	verb	{approved, fly, . . . }	verb the PP attaches to
17	verb-lemma	{approve, be, . . . }	verb lemma the PP attaches to
18	**verb-pas**	{yes, no}	is verb in passive voice?

Table 5: Features used for extracting MANNER from PPs, their values and explanation. Features in bold letters are new and specialized on detecting MANNER from PPs.

CoNLL-2005 Shared Task (Carreras and Màrquez, 2005), combined with new, manner-specific features that we introduce. These new features bring a significant improvement and are dependent on the phrase potentially encoding a MANNER. Experimentation has shown that MANNER relations expressed by an ADVP are easier to detect than the ones expressed by a PP.

Adverbial Phrases The feature set used is depicted in Table 4. Some features are typical of semantic role labeling, but features adverb, dictionary and ends-with-ly are specialized to MANNER extraction from ADVPs. These three additional features bring a significant improvement (Section 7). We only provide details for the non-obvious features.

The main adverb and verb are retrieved by selecting the last adverb or verb of a sequence. For example, in *more strongly*, the main adverb is *strongly*, and in *had been rescued* the main verb is *rescued*.

Dictionary tests the presence of the adverb in a custom built dictionary which contains all lemmas for adverbs in WordNet whose gloss matches the regular expression *in a .* (manner|way|fashion|style)*. For example, *more.adv.1: used to form the comparative of some adjectives and adverbs* does not belong to the dictionary, and *strongly.adv.1: with strength or in a* *strong manner* does. This feature is an extension of the dictionary presented in (Girju et al., 2003).

Given the sentence *[...] We [work [damn hard]$_{ADVP}$ at what we do for damn little pay]$_{VP}$, and [...]* (wsj_1144, 128), the features are: {parent-node:VP, num-leaves:2, adverb:hard, dictionary:no, ends-with-ly:no, POS-tag-bef:RB, POS-tag-aft:IN, verb:work, distance:1}, and it is a positive instance.

Prepositional Phrases PPs are known to be highly ambiguous and more features need to be added. The complete set is depicted in Table 5.

Some features are typical of semantic role detection; we only provide a justification for the new features added. `Num-pp-bef` and `num-pp-aft` captures the number of PP siblings before and after the PP. The relative order of PPs is typically MANNER, AT-LOCATION and AT-TIME (Hawkins, 1999), and this feature captures this idea without requiring temporal or local annotation.

PPs having quotes are more likely to encode a MANNER, the chunk of text between quotes being the manner. For example, *use in "very modest amounts"* (wsj_0003, 10) and *reward with "page bonuses"* (wsj_0012, 8).

`head-np` indicates the head noun of the NP that attaches to the preposition to form the PP. It is retrieved by selecting the last noun in the NP. Certain nouns are more likely to indicate a MANNER than others. This feature captures the domain restriction. For nouns, only non temporal abstract objects and states can encode a MANNER. Some examples of positive instances are *haul in the guests' [honor]*, *lift in two [stages]*, *win at any [cost]*, *plunge against the [mark]* and *ease with little [fanfare]*. However, counterexamples can be found as well: *say through his [spokesman]* and *do over the [counter]*.

`Verb-pas` indicates if a verb is in passive voice. In that case, a PP starting with *by* is much more likely to encode an AGENT than a MANNER. For example, compare (1) *"When the fruit is ripe, it [falls]$_y$ from the tree [by itself]$_{PP}$," he says.* (wsj_0300, 23); and (2) *Four of the planes [were purchased]$_y$ [by International Lease]$_{PP}$ from Singapore Airlines in a [...] transaction* (wsj_0243, 3). In the first example a MANNER holds; in the second an AGENT.

Given the sentence *Kalipharma is a New Jersey-based pharmaceuticals concern that [sells products [under the Purepac label]$_{PP}$]$_{VP}$.* (wsj_0023, 1), the features are: {parent-node:VP, next-node:-, num-pp-bef:0, num-pp-aft:0, first-word:under, first-POS-tag:IN, first-prep:under, POS-tag-bef:NNS, POS-tag-aft:DT, word-aft:the, has-rb:no, has-quotes:no, head-np-lemma:label, head-is-last:yes, head-has-cap:yes, verb:sells, verb-lemma:sell, verb-pas:no}, and it is a positive instance.

7 Learning Algorithm and Results

7.1 Experimental Results

As a learning algorithm we use a Naive Bayes classifier, well known for its simplicity and yet good performance. We trained our models with the training corpus using 10-fold cross validation, and used the held-out portion to tune the feature set and adjust parameters. More features than the ones depicted were tried, but we only report the final set. For example, named entity recognition and flags indicating the presence of AT-LOCATION and AT-TIME relations for the verb were tried, but they did not bring any significant improvement.

Table 6 summarizes the results obtained. We report results only on the test corpus, which corresponds to instances not seen before and therefore they are a honest estimation of the performance. The improvement brought by subsets of features and statistical significance tests are also reported. We test the significance of the difference in performance between two feature sets i and j on a set of ins instances with the Z-score test, where $z = \frac{abs(err_i, err_j)}{\sigma_d}$, err_k is the error made using set k, and $\sigma_d = \sqrt{\frac{err_i(1-err_i)}{ins} + \frac{err_j(1-err_j)}{ins}}$.

ADVPs The full set of features yields a F-measure of 0.815. The three specialized features (3, 4 and 5) are responsible for an improvement of .168 in the F-measure. This difference in performance yields a Z-score of 7.1, which indicates that it is statistically significant.

PPs All the features proposed yield a F-measure of 0.693. The novel features specialized in MANNER detection from PPs (in bold letters in Table 5) bring an improvement of 0.059, which again is significant.

Phrase	#MNR	Feat. Set	#MNR retrieved	#MNR correct	P	R	F
ADVP	678	1,2,6-9	908	513	.565	.757	.647
		all	757	585	.773	.863	**.815**
PP	705	1,2,5,6,8-10,16,17	690	442	.641	.627	.634
		all	713	491	.689	.696	**.693**

Table 6: Results obtained during testing for different sets of features.

The Z-score is 2.35, i.e., the difference in performance is statistically significant with a confidence greater than 97.5%. Thus, adding the specialized features is justified.

7.2 Error Analysis

The mapping of words to semantic classes is data-driven and decisions were taken so that the overall accuracy is high. However, mistakes are made. Given *We want to [see]*$_y$ *the market from the inside*, the underlined PP encodes a MANNER and the mapping proposed (Table 1) does not map *inside* to ntao. Similarly, given *Like their cohorts in political consulting, the litigation advisers [encourage]*$_y$ *their clients [. . .]*, the underlined text encodes a MANNER and yet *cohorts* is subsumed by *social_group.n.1* and therefore is not mapped to ntao.

The model proposed for MANNER detection makes mistakes as well. For ADVPs, if the main adverb has not been seen during training, chances of detecting MANNER are low. For example, the classifier fails to detect the following MANNER relations: *[. . .] which together own about [. . .]* (wsj_0671, 1); and *who has ardently supported [. . .]* (wsj_1017, 26) even though *ardently* is present in the dictionary and ends in *-ly*;

For PPs, some errors are due to the PropBank annotation. For example, in *Shearson Lehman Hutton began its coverage of the company with favorable ratings.* (wsj_2061, 57), the underlined text is annotated as ARG2, even though it does encode a MANNER. Our model correctly detects a MANNER but it is counted as a mistake. Manners encoded by *under* and *at* are rarely detected, as in *that have been consolidated in federal court under U.S. District Judge Milton Pollack* (wsj_1022.mrg, 10).

8 Comparison with Previous Work

To the best of our knowledge, there have not been much efforts to detect MANNER alone. Girju et al. (2003), present a supervised approach for ADVP similar to the one reported in this paper, yielding a F-measure of .665. Our augmented feature set obtains a F-measure of .815, clearly outperforming their method (Z-test, confidence $>$ 97.5%). Moreover, ADVPs only represent 45.3% of MANNER as a semantic role in PropBank. We also have presented a model to detect MANNER encoded by a PP, the other big chunk of MANNER (44.6%) in PropBank.

Complete systems for Semantic Role Labeling perform poorly when detecting MANNER; the Top-10 systems in CoNLL-2005 shared task[3] obtained F-measures ranging from .527 to .592. We have trained our models using the training data provided by the task organizers (using the Charniak parser syntactic information), and tested with the provided test set (test.wsj). Our models yield a Precision of .759 and Recall of .626 (F-measure .686), bringing a significant improvement over those systems (Z-test, confidence $>$ 97.5%). When calculating recall, we take into account all MANNER in the test set, not only ADVPs and PPs whose fathers are S or VP (i.e. not only the ones our models are able to detect). This evaluation is done with exactly the same data provided from the task organizers for both training and test.

Unlike typical semantic role labelers, our features do not include rich syntactic information (e.g. syntactic path from verb to the argument). Instead, we only require the value of the parent and in the case of PPs, the sibling node. When repeating the CoNLL-2005 Shared Task training and test using gold syntactic information, the F-measure obtained is .714, very close to the .686 obtained with Charniak syntactic trees (not significant, confidence $>$ 97.5%).

[3] http://www.lsi.upc.es/˜srlconll/st05/st05.html

Even though syntactic parsers achieve a good performance, they make mistakes and the less our models rely on them, the better.

9 Composing MANNER with PURPOSE

MANNER can combine with other semantic relations in order to reveal implicit relations that otherwise would be missed. The basic idea is to compose MANNER with other relations in order to infer another MANNER. A necessary condition for combining MANNER with another relation R is the compatibility of RANGE(MNR) with DOMAIN(R) or RANGE(R) with DOMAIN(MNR). The extended definition (Section 3) allows to quickly determine if two relations are compatible (Blanco et al., 2010).

The new MANNER is automatically inferred by humans when reading, but computers need an explicit representation. Consider the following example: *[. . .] the traders [place]*$_{y}$ *orders [via computers]*$_{MNR}$ *[to buy the basket of stocks . . .]*$_{PRP}$ (wsj_0118, 48). PropBank states the basic annotation between brackets: *via computers* is the MANNER and *to buy the basket [. . .]* the PURPOSE of the *place orders* event. We propose to combine these two relations in order to come up with the new relation MNR(*via computers*, *buy the basket [. . .]*). This relation is obvious when reading the sentence, so it is omitted by the writer. However, any semantic representation of text needs as much semantics as possible explicitly stated.

This claim is supported by several PropBank examples: (1) *The classics have [zoomed]*$_{y}$ *[in price]*$_{MNR}$ *[to meet the competition]*$_{PRP}$*, and . . .* (wsj_0071, 9) and (2) *. . . the government [curtailed]*$_{y}$ *production [with land-idling programs]*$_{MNR}$ *[to reduce price-depressing surpluses]*$_{PRP}$ (wsj_0113, 12). In both examples, PropBank encodes the MANNER and PURPOSE for event y indicated with brackets. After combining both relations, two new MANNER arise: MNR(*in price*, *meet the competition*) and MNR(*with land-idling programs*, *reduce price-depressing surpluses*).

Out of 237 verbs having in PropBank both PURPOSE and MANNER annotation, the above inference method yields 189 new valid MANNER not present in PropBank (Accuracy .797).

MANNER and other relations. MANNER does not combine with relations such as CAUSE, AT-LOCATION or AT-TIME. For example, given *And they continue [anonymously]*$_{x,MNR}$ *[attacking]*$_{y}$ *CIA Director William Webster [for being too accommodating to the committee]*$_{z,CAU}$ (wsj_0590, 27), there is no relation between x and z. Similarly, given *[In the tower]*$_{x,LOC}$*, five men and women [pull]*$_{y}$ *[rhythmically]*$_{z,MNR}$ *on ropes attached to [. . .]* (wsj_0089, 5) and *[In May]*$_{x,TMP}$*, the two companies, [through their jointly owned holding company]*$_{z,MNR}$*, Temple, [offered]*$_{y}$ *[. . .]* (wsj_0063, 3), no connection exists between x and z.

10 Conclusions

We have presented a supervised method for the automatic discovery of MANNER. Our approach is simple and outperforms previous work. Our models specialize in detecting the most common pattern encoding MANNER. By doing so we are able to specialize our feature sets and outperform previous approaches that followed the idea of using dozens of features, most of them potentially useless, and letting a complicated machine learning algorithm decide the actual useful features.

We believe that each relation or role has its own unique characteristics and capturing them improves performance. We have shown this fact for MANNER by examining examples, considering the kind of arguments that can be part of the domain and range, and considering theoretical works (Hawkins, 1999).

We have shown performance using both gold syntactic trees and the output from the Charniak parser, and there is not a big performance drop. This is mainly due to the fact that we do not use deep syntactic information in our feature sets.

The combination of MANNER and PURPOSE opens up a novel paradigm to perform semantic inference. We envision a layer of semantics using a small set of basic semantic relations and inference mechanisms on top of them to obtain more semantics on demand. Combining semantic relations in order to obtain more relation is only one of the possible inference methods.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 17th international conference on Computational Linguistics*, Montreal, Canada.

Eduardo Blanco, Hakki C. Cankaya, and Dan Moldovan. 2010. Composition of Semantic Relations: Model and Applications. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, Beijing, China.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: semantic role labeling. In *CONLL '05: Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 152–164, Morristown, NJ, USA.

Du S. Chang and Key S. Choi. 2006. Incremental cue phrase learning and bootstrapping method for causality extraction using cue phrase and word pair probabilities. *Information Processing & Management*, 42(3):662–678.

Dmitry Davidov and Ari Rappoport. 2008. Unsupervised Discovery of Generic Relationships Using Pattern Clusters and its Evaluation by Automatically Generated SAT Analogy Questions. In *Proceedings of ACL-08: HLT*, pages 692–700, Columbus, Ohio.

Dmitry Davidov, Ari Rappoport, and Moshe Koppel. 2007. Fully Unsupervised Discovery of Concept-Specific Relationships by Web Mining. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 232–239, Prague, Czech Republic.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic Labeling Of Semantic Roles. *Computational Linguistics*, 28:245–288.

Roxana Girju, Manju Putcha, and Dan Moldovan. 2003. Discovery of Manner Relations and Their Applicability to Question Answering. In *Proceedings of the ACL 2003 Workshop on Multilingual Summarization and Question Answering*, pages 54–60, Sapporo, Japan.

Roxana Girju, Adriana Badulescu, and Dan Moldovan. 2006. Automatic Discovery of Part-Whole Relations. *Computational Linguistics*, 32(1):83–135.

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 the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 13–18, Prague, Czech Republic.

Ana M. Giuglea and Alessandro Moschitti. 2006. Semantic role labeling via FrameNet, VerbNet and PropBank. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 929–936, Morristown, NJ, USA.

John A. Hawkins. 1999. The relative order of prepositional phrases in English: Going beyond Manner-Place-Time. *Language Variation and Change*, 11(03):231–266.

Marti A. Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*, pages 539–545.

Hermann Helbig. 2005. *Knowledge Representation and the Semantics of Natural Language*. Springer.

Toru Hirano, Yoshihiro Matsuo, and Genichiro Kikui. 2007. Detecting Semantic Relations between Named Entities in Text Using Contextual Features. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, Demo and Poster Sessions*, pages 157–160, Prague, Czech Republic.

Lluís Màrquez, Xavier Carreras, Kenneth C. Litkowski, and Suzanne Stevenson. 2008. Semantic Role Labeling: An Introduction to the Special Issue. *Computational Linguistics*, 34(2):145–159.

Dan Moldovan, Christine Clark, and Mitchell Bowden. 2007. Lymba's PowerAnswer 4 in TREC 2007. In *Proceedings of the Sixteenth Text REtrieval Conference (TREC 2007)*.

Paul Nulty. 2007. Semantic Classification of Noun Phrases Using Web Counts and Learning Algorithms. In *Proceedings of the ACL 2007 Student Research Workshop*, pages 79–84, Prague, Czech Republic.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106.

Vivek Srikumar, Roi Reichart, Mark Sammons, Ari Rappoport, and Dan Roth. 2008. Extraction of Entailed Semantic Relations Through Syntax-Based Comma Resolution. In *Proceedings of ACL-08: HLT*, pages 1030–1038, Columbus, Ohio.

Barker Szpakowicz, Ken Barker, and Stan Szpakowicz. 1995. Interactive semantic analysis of Clause-Level Relationships. In *Proceedings of the Second Conference of the Pacific Association for Computational Linguistics*, pages 22–30.

Marta Tatu. 2005. Automatic Discovery of Intentions in Text and its Application to Question Answering. In *Proceedings of the ACL Student Research Workshop*, pages 31–36, Ann Arbor, Michigan.

Peter D. Turney. 2006. Expressing Implicit Semantic Relations without Supervision. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 313–320, Sydney, Australia.

Tense Sense Disambiguation: a New Syntactic Polysemy Task

Roi Reichart
ICNC
Hebrew University of Jerusalem
roiri@cs.huji.ac.il

Ari Rappoport
Institute of Computer Science
Hebrew University of Jerusalem
arir@cs.huji.ac.il

Abstract

Polysemy is a major characteristic of natural languages. Like words, syntactic forms can have several meanings. Understanding the correct meaning of a syntactic form is of great importance to many NLP applications. In this paper we address an important type of syntactic polysemy – the multiple possible senses of tense syntactic forms. We make our discussion concrete by introducing the task of *Tense Sense Disambiguation* (TSD): given a concrete tense syntactic form present in a sentence, select its appropriate sense among a set of possible senses. Using English grammar textbooks, we compiled a syntactic sense dictionary comprising common tense syntactic forms and semantic senses for each. We annotated thousands of BNC sentences using the defined senses. We describe a supervised TSD algorithm trained on these annotations, which outperforms a strong baseline for the task.

1 Introduction

The function of syntax is to combine words to express meanings, using syntactic devices such as word order, auxiliary words, and morphology (Goldberg, 1995). Virtually all natural language devices used for expressing meanings (e.g., words) exhibit polysemy. Like words, concrete syntactic forms (the sentence words generated by specific syntactic devices) can have several meanings. Consider the following sentences:

(a) They **are playing** chess in the park.

(b) They **are playing** chess next Tuesday.

Both contain the concrete syntactic form 'are playing', generated by the abstract syntactic form usually known as 'present progressive' (am/is/are + V-ing). In (a), the meaning is 'something happening now', while in (b) it is 'a plan to do something in the future'. Note that the polysemy is of the syntactic form as a unit, not of individual words. In particular, the verb 'play' is used in the same sense in both cases.

In this paper we address a prominent type of syntactic form polysemy: the multiple possible senses that tense syntactic forms can have. Disambiguating the polysemy of tense forms is of theoretical and practical importance (Section 2). To make our discussion concrete, we introduce the task of *Tense Sense Disambiguation* (TSD): given a concrete tense syntactic form in a sentence, select its correct sense among a given set of possible senses (Section 3).

The disambiguation of polysemy is a fundamental problem in NLP. For example, Word Sense Disambiguation (WSD) continues to attract a large number of researchers (Agirre and Edmonds, 2006). TSD has the same structure as WSD, with different disambiguated entities.

For experimenting with the TSD task, we compiled an English syntactic sense dictionary based on a thorough study of three major English grammar projects (Section 4). We selected 3000 sentences from the British National Corpus containing 4702 concrete syntactic forms, and annotated each of these by its sense (Section 5).We developed a supervised learning TSD algorithm that uses various feature types and takes advantage of the task structure (Section 6). Our algorithm substantially outper-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 325–334,
MIT, Massachusetts, USA, 9-11 October 2010.

forms the ‘most frequent sense’ baseline (Section 7).

TSD is fundamental to sentence understanding and thus to NLP applications such as textual inference, question answering and information retrieval. To the best of our knowledge, this is the first paper to address this task. In Section 8 we discuss research directions relevant to TSD placing the new task in the context of the previous research of syntactic ambiguity resolution.

2 TSD Motivation

In this work we follow linguistics theories that posit that tense does not directly reflect conceptual time as one might think. Dinsmore (1991) and Cutrer (1994) explain that the same tense may end up indicating very different objective time relations relative to the sentence production time.

Fauconnier (2007) exemplifies such phenomena. In the following sentences, the present tense corresponds to the *future* time: (1) The boat leaves next week. (2) When he comes tomorrow, I will tell him about the party. (3) If I see him next week, I will ask him to call you.

In contrast, the following present tense sentences talk about events that happened in the *past*: (1) I am walking down the street one day when suddenly this guy walks up to me. (2) He catches the ball. He runs. He makes a touchdown. (morning-after sports report).

Another set of examples is related to the past tense. In the following sentences it corresponds to a *present* time: (1) Do you have a minute? I wanted to ask you a question. (2) I wish I lived closer to my family now. In contrast, in the following two sentences, it corresponds to a future time: (1) If I had the time next week, I would go to your party. (2) I cannot go to the concert tonight. You will have to tell me how it was.

Fauconnier explains these phenomena by a model for the grammar of tense. According to this model, the grammar specifies partial constraints on time and fact/prediction status that hold locally between mental spaces within a discourse configuration. We may obtain actual information about time by combining this with other available pragmatic information. Accordingly, the same tense may end up indicating very different objective time relations relative to the speech event.

TSD fits well with modern linguistics theories. For example, in the construction grammar framework (Goldberg, 1995), the ‘construction’ is the basic unit, comprised of a form and a meaning. Words, multiword expressions, and syntactic forms are all valid constructions. It is thus very natural to address the sense disambiguation problem for all of these. In this paper we focus on tense constructions.

For many NLP applications, it is very important to disambiguate the tense forms of the sentence. Among these applications are: (1) machine translation, as the actual time described by one tense form in the source language may be described by a different tense form in the target language; (2) understanding the order of events in a text; (3) textual entailment, when the optional entailed sentences refer to the time and/or order of events of the source sentence. Many more examples also exist.

3 The TSD Task

In this section we formally define the TSD task, discuss its nature vs. WSD, and describe various concrete task variants.

Task definition. First, some essential terminology. The function of syntax is to combine lexical items (words, multiword expressions) to express meanings. This function is achieved through syntactic *devices*. The most common devices in English are word order, morphology, and the usage of auxiliary words. An *Abstract Syntactic Form (ASF)* is a particular set of devices that can be used to express a set of meanings. A *Concrete Syntactic Form (CSF)* is a concrete set of words generated by an ASF for expressing a certain meaning in an utterance[1]. A CSF is *ambiguous* if its generating ASF has more than one meaning, which is the usual case. In this case we also say that the ASF is ambiguous.

Here are a few examples. The ‘present progressive’ ASF has the form ‘am/is/are V-ing’[2], which employs all three main devices. It is ambiguous,

[1] In some linguistic theories, the central notion is the *construction*, which combines an ASF (referred to as the form of the construction) with a single meaning (Goldberg, 1995).

[2] Note that strictly speaking, these are three different ASFs. We refer to this ASF family by a single name because they have the same set of meanings and because it is standard to treat them as a single ASF.

as shown in Section 1. The 'present simple' ASF has the form 'V(+s)'[3], and is ambiguous as well: in the sentence 'My Brother arrives this evening', the CSF 'arrives' conveys the meaning of 'a future event arranged for a definite time', while in the sentence 'The sun rises in the East' the meaning is that of a repeated event.

TSD vs. WSD. The TSD task is to disambiguate the semantic sense of a tense syntactic form. TSD is clearly different from WSD. This is obvious when the CSF comprises two words that are not a multiword expression, and is usually also the case when it comprises a single word. Consider the 'My Brother arrives this evening' example above. While the verb 'arrive' has two main senses: 'reach a place', and 'begin', as in 'Summer has arrived', in that example we focused on the disambiguation of the tense sense of the 'arrives' construction.

Concrete task variants. Unlike with words, the presence of a particular CSF in a sentence is not trivially recognizable. Consequently, there are three versions of the TSD task: (1) we are given the sentence, a marked subset of its words comprising a CSF, and the ASF that has generated these words; (2) we are given the sentence and a marked subset of its words comprising a CSF, without knowing the generating ASF; (3) we are given only the sentence and we need to find the contained CSFs and their ASFs. In all cases, we need to disambiguate the sense of the ASFs. We feel that the natural granularity of the task is captured by version (2). However, since the ASF can usually be identified using relatively simple features, we also report results for version (1). The main difficulty in all versions is identifying the appropriate sense, as is the case with WSD.

4 The Syntactic Sense Dictionary

A prerequisite to any concrete experimentation with the TSD task is a syntactic sense dictionary. Based on a thorough examination of three major English grammar projects, we compiled a set of 18 common English tense ASFs and their possible senses. The projects are (1) the Cambridge University Press English Grammar In Use series, comprising three books (essential, intermediate and advanced) (Murphy, 2007; Murphy, 1994; Hewings, 2005); (2) the English grammar texts resulting from the seminal corpus-based Cobuild project (elementary, advanced) (Willis and Wright, 2003; Willis, 2004); (3) the Longman Grammar of Spoken and Written English (Biber et al., 1999).

As in any sense dictionary, in many cases it is hard to draw the line between senses. In order to be able to explore the computational limits of the task, we have adopted a policy of fine sense granularity. For example, senses 1 and 3 of the 'present simple' ASF in Table 1 can be argued to be quite similar to each other, having a very fine semantic distinction. A specific application may choose to collapse some senses into one.

We used the conventional ASF names, which should not be confused with their meanings (e.g., the 'present simple' ASF can be used to refer to *future*, not present, events, as in Table 1, sense 4).

The ASF set thus obtained is: real conditionals, hypothetical conditionals, wishes, reported speech, present simple, present progressive, present perfect, present perfect progressive, past simple, past progressive, past perfect, past perfect progressive, 'be + going + to + infinitive', future progressive, future perfect, future perfect progressive, 'would' tense forms, and 'be + to + infinitive'. Note that the first four ASFs are not direct tense forms; we include them because they involve tensed sub-sentences whose disambiguation is necessary for disambiguation of the whole ASF. The total number of possible senses for these 18 ASFs is 103.

Table 1 shows the complete senses set for the 'present simple' and 'be + to + infinitive' ASFs, plus an example sentence for each sense. Space limitations prevent us from listing all form senses here; we will make the listing available online.

5 Corpus Creation and Annotation

We selected 3000 sentences from the British National Corpus (BNC) (Burnard, 2000), containing 4702 CSFs (1.56 per sentence). These sentences with their CSFs were sense annotated. To select the 3000 sentences, we randomly sampled sentences from the various written and spoken sections of the

[3] Again, these are two ASFs, one adding an 's' and one using the verb as is.

	Present Simple
1	*Things that are always true* It gets cold in the winter.
2	*Regular and repeated actions and habits* My parents often eat meat.
3	*General facts* Mr. Brown is a teacher.
4	*A future event arranged for a definite time* The next train arrives at 11:30.
5	*Plans, expectations and hopes* We hope to see you soon.
6	*Ordering someone to do something* Take your hands out of your pockets!
7	*Something happening now, with verbs that are not used in the present progressive in this sense* I do not deny the allegation.
8	*Events happening now (informal; common in books, scripts, radio etc.)* She goes up to this man and looks into his eyes.
9	*Past actions* I was sitting in the park reading a newspaper when all of a sudden this dog jumps at me.
10	*Newspaper headlines, for recent events* Quake hits central Iran.
11	*When describing the content of a book* Thompson gives an exhaustive list in chapter six.
	'be + to + infinitive'
1	*Events that are likely to happen in the near future* Police officers are to visit every home in the area.
2	*Official arrangements, formal instructions & orders* You are not to leave without my permission.
3	*In an if-clause to say that something must happen before something else can happen* If the human race is to survive, we must look at environmental problems now.

Table 1: The full set of senses of the 'present simple' and 'be + to + infinitive' abstract syntactic forms (ASFs), with an example for each.

corpus, giving each section an equal weight. To guarantee ample representation of ASFs, we manually defined auxiliary words typical of each ASF (e.g., 'does', 'been' etc), and sampled hundreds of sentences for each set of these auxiliary words. To make sure that our definition of auxiliary words does not skew the sampling process, and to obtain ASFs that do not have clear auxiliary words, we have also added 1000 random sentences. The number of CSF instances obtained for each ASF ranges from 100 (future perfect) to over 850 (present simple). All senses are represented; the number of senses represented by at least 15 CSFs is 77 (out of 103, average number of CSFs per sense is 45.65).

We implemented an interactive application that displays a sentence and asks an annotator to (1) mark words that participate in the CSFs contained in the sentence; (2) specify the ASF(s) of these CSFs; and (3) select the appropriate ASF sense from the set of possible senses. Annotators could also indicate 'none of these senses', which they did for 2.6% (122 out of 4702) of the CSFs.

Annotation was done by two annotators (university students). To evaluate inter-annotator agreement, a set of 210 sentences (7% of the corpus), containing at least 10 examples of each ASF, was tagged by both annotators. The CSF+ASF identification inter-annotator agreement was 98.7%, and the inter-annotator agreement for the senses was 84.2%. We will make the annotated corpus and annotation guidelines available online.

6 Learning Algorithm

In this section we describe our learning model for the TSD task. First, note that the syntactic sense is not easy to deduce from readily computable annotations such as the sentence's POS tagging, dependency structure, or parse tree (see Section 8). Hence, a learning algorithm is definitely needed.

As common in supervised learning, we encode the CSFs into feature vectors and then apply a learning algorithm to induce a classifier. We first discuss the feature set and then the algorithm.

Features. We utilize three sets of features: basic features, lexical features, and a set of features based on part-of-speech (POS) tags (Table 2). The 'auxiliary words' referred to in the table are the manually specified words for each ASF that have assisted us in sampling the corpus (see Section 5). 'Content words' are the non-auxiliary words appearing in the CSF[4]. Content words are usually verbs, since we focus here on tense-related ASFs. The position and distance of a form are based on its leftmost word (auxiliary or content).

The personal pronouns used in the position features are: I, you, he, she, it, they, and we. For

[4] Usually, there is a single content word. However, there may be more than one, e.g. for phrasal verbs.

simplicity, we considered every word starting with a capital letter that is not the first word in the sentence to be a name.

Each ‘Conditional’ CSF contains two tense CSFs. The one that is not the CSF currently encoded by the features is referred to as its ‘mate’.

For the time lexical features we used 16 words (e.g., recently, often, now). For the reported speech lexical features we used 14 words (e.g., said, replied, wrote[5]). The words were obtained from the grammar texts and our corpus development set.

The POS tagset used by the POS-based features is that of the WSJ PennTreebank (see Section 7). The possible verb tags in this tagset are: VB for the base form, VBD for past tense, VBN for past participle, VBG for a present participle or gerund (-ing), VBP for present tense that is not 3rd person singular, and VBZ for present simple 3rd person singular.

Conjunctions and prepositions are addressed through the POS tags CC and IN. Using the PRP tag to detect pronouns or lexical lists for conjunctions and prepositions yielded no significant change in the results.

In Section 7 we explore the impact each of the feature sets has on the performance of the algorithm. Our results indicate that the basic features have the strongest impact, the POS-based features enhance the performance in specific cases and the lexical features only marginally affect the final results.

Algorithm. Denote by x_i the feature vector of a CSF instance i, by C_i the set of possible labels for x_i, and by $c_i \in C_i$ the correct label. The training set is $\{(x_j, C_j, c_j)\}_{j=1}^{n}$. Let (x_{n+1}, C_{n+1}) be a test CSF. As noted in Section 3, there are two versions of the task, one in which C_i includes the totality of sense labels, and one in which it includes only the labels associated with a particular ASF. In both cases, the task is to select which of the labels in C_{n+1} is its correct label c_{n+1}.

Owing to the task structure, it is preferable to use an algorithm that allows us to restrict the possible labels of each CSF. For both task versions, this would help in computing better probabilities during the training stage, since we know the ASF type of training CSFs. For the task version in which the ASF type is known at test time, this would also help during the test stage.

Basic Features
Form words. Auxiliary and content words of the CSF.
Form type. The type, if it is known during test time.
Other forms. The auxiliary and content words (and type, if known) of the other CSFs present in the sentence.
Position. The position of the CSF in the sentence, its distance from the end of the sentence, whether it is in the first (last) three words in the sentence, its distance from the closest personal pronoun or name.
Wish. Is there a CSF of type ‘wish’ before the encoded form, the number of CSFs between that ‘wish’ form and the encoded CSF (if there are several such ‘wish’ forms, we take the closest one to the encoded form).
Conditional. Does the word ‘if’ appear before the encoded form, is the ‘if’ the first word in the sentence, the number of CSFs between the ‘if’ and the encoded form, the auxiliary and content words (and type, if known) of the mate form, is there a comma between the encoded form and its mate form, does the word ‘then’ appear between the encoded form and its mate form.
Punctuation. The type of end of sentence marker, distance of the encoded form from the closest predecessor (successor) comma.
Lexical Features
Time. Time words appearing in the sentence, if any.
Reported speech. Reported speech words appearing in the sentence, if any.
Be. Does the encoded form contain the verb ‘be’.
Features Based on POS Tags
*Form.*The POS of the verb in the encoded form.
Other forms. The POS of the verb in the other CSFs in the sentence.
POS tags. The POS tags of the two words to the left (right) of the encoded form.
Conjunction POS. Is there a Conjunction (CC) between the encoded form and its closest predecessor (successor) form, the distance from that conjunction.
Preposition POS. Is there a Preposition (IN) between the encoded form and its closest predecessor (successor) form, the distance from that preposition.

Table 2: Basic features (top), lexical features (middle) and POS tags-based features (bottom) used by the TSD classifier.

For the version in which ASF type is known at test time, we experimented in two scenarios. In the first,

[5] These are all in a past form due to the semantics of the reported speech form.

we take the ASF type at test time from the manual annotation and provide it to the algorithm. In the second, instead of the manual annotation, we implemented a simple rule-based classifier for selecting ASF types. The classifier decides what is the type of an ASF according to the POS tag of its verb and to its auxiliary words (given in the annotation). For example, if we see the auxiliary phrase 'had been' and the verb POS is not VBG, then the ASF is 'past perfect simple'. This classifier's accuracy on our development (test) data is 94.1 (91.6)%. In this scenario, when given a test CSF, X_{n+1}, its set of possible labels C_{n+1} is defined by the classifier output. In the features in which ASF type is used (see table 2), it is taken from the classifier output in this case.

The sequential model algorithm presented by Even-Zohar and Roth (2001) directly supports this label restriction requirement [6]. We use the SNOW learning architecture for multi-class classification (Roth, 1998), which contains an implementation of that algorithm. The SNOW system allows us not to define restrictions if so desired. It also lets us choose the learning algorithm used when it builds its classifier network. The algorithm can be Perceptron (MacKay, 2002), Winnow (Littlestone, 1988) or Naive Bayes (MacKay, 2002)[7]. In Section 7 we analyze the effect that these decisions have on our results.

Classifier Selection. Investigating the best configuration of the SNOW system with development data, we found that Naive Bayes gave the best or close to best result in all experimental conditions. We therefore report our results when this algorithm is used. Naive Bayes is particularly useful when relatively small amounts of training CSF instances are available (Zhang, 2004), and achieves good results when compared to other classifiers for the WSD task (Mooney, 1996), which might explain our results. Fine tuning of Winnow parameters also leads to high performance (sometimes the best), but most other parameter configurations lead to disappointing results. For the Perceptron, most parameter configurations lead to good results (much better than the baseline), but these were a few percent worse than the best Winnow or Naive Bayes results.

7 Experimental Results

Experimental setup. We divided the 3000 annotated sentences (containing 4702 CSFs) to three datasets: training data (2100 sentences, 3183 forms), development data (300 sentences, 498 forms) and test data (600 sentences, 1021 forms). We used the development data to design the features for our learning model and to tune the parameters of the SNOW sequential model. In addition we used this data to design the rules of the ASF type classifier (which is not statistical and does not have a training phase).

For the POS features, we induced POS tags using the MXPOST POS tagger (Ratnaparkhi, 1996). The tagger was trained on sections 2-21 of the WSJ PennTreebank (Marcus et al., 1993) annotated with gold standard POS tags. We used a publicly available implementation of the sequential SNOW model[8].

We experimented in three conditions. In the first (TypeUnknown), the ASF type is not known at test time. In the last two, it is known at test time. These two conditions differ in whether the type is taken from the gold standard annotation of the test sentences (TypeKnown), or from the output of the simple rule-based classifier (TypeClassifier, see Section 6). For both conditions, the results reported below are when both ASF type features and possible labels sets are provided during training by the manual annotation. This is true also for the training of the MFS baseline (see below)[9].

We report an algorithm's quality using accuracy, that is, the number of test CSFs that were correctly resolved by the algorithm divided by the total number of test CSFs.

Baseline. We compared the performance of our algorithm to the 'most frequent sense' (MFS) base-

[6] Note that the name of the learning algorithm is derived from the fact that it utilizes classifiers to sequentially restrict the number of competing classes while maintaining with high probability the presence of the true outcome. The classification task it performs is not sequential in nature.

[7] Or a combination of these algorithms, which we did not explore in this paper.

[8] http://l2r.cs.uiuc.edu/∼cogcomp/asoftware.php?skey=SNOW

[9] For the TypeClassifier condition, we also experimented using an ML technique that sometimes reduces noise, where training is done using the classifier types. We obtained very similar results to those reported.

	TypeUnknown	TypeClassifier	TypeKnown
Our algorithm	**49.7%**	**58.8%**	**62%**
MFS baseline	13.5%	42.9%	46.7%

Table 3: Performance of our algorithm and of the MFS baseline where at test time ASF type is known (right), unknown (left) or given by a simple rule-based classifier (middle). Our algorithm is superior in all three conditions.

	Constrained Model		Unconstrained Classifier	
	All features	Base+Lexical features	All features	Base+Lexical features
Type features	57.9%	57.7%	53%	50.1%
No type features	57.2%	55.4%	48%	42.6%

Table 4: Impact of POS features. When the constrained model is used (left section), POS features have no effect on the results when ASF type information is encoded. When an unconstrained classifier is used, POS features affect the results both when ASF type features are used and when they are not (see discussion in the text).

line. This baseline is common in semantic disambiguation tasks and is known to be quite strong. In the condition where the ASF type is not known at test time, MFS gives each form in the test set the sense that was the overall most frequent in the training set. That is, in this case the baseline gives all test set CSFs the same sense. When the ASF type is known at test time, MFS gives each test CSF the most frequent sense *of that ASF type* in the training set. That is, in this case all CSFs having the same ASF type get the same sense, and forms of different types are guaranteed to get different senses.

Recall that the condition where ASF type is known at test time is further divided to two conditions. In the TypeKnown condition, MFS selects the most frequent sense of the manually created ASF type, while in the TypeClassifier condition it selects the most frequent sense of the type decided by the rule-based classifier. In this condition, if the classifier makes a mistake, MFS will necessarily make a mistake as well.

Note that a random baseline which selects a sense for every test CSF from a uniform distribution over the possible senses (103 in our case) would score very poorly.

Results. Table 3 shows our results. Results are shown where ASF type is not known at test time (left), when it is decided at test time by a rule-based classifier (middle) and when it is known at test time (right). Our algorithm outperforms the MFS baseline in all three conditions. As expected, both our algorithm and the MFS baseline perform better when ASF type information is available at test time (TypeClassifier and TypeKnown conditions), and improve as this data becomes more accurate (the TypeKnown condition)[10].

Analyzing the per-type performance of our algorithm reveals that it outperforms the MFS baseline for each and every ASF type. For example, in the TypeKnown condition, the accuracy gain of our algorithm over the baseline[11] varies from 4% for the ‘present perfect’ to 30.6% and 29.1% for the ‘past perfect’ and ‘present simple’ ASFs.

Below we analyze the roles of the different components of our learning algorithm in performing the TSD task. Since this is the first exploration of the task, it is important to understand what properties are essential for achieving good performance. The analysis is done by experimenting with development data, and focuses on the TypeKnown and TypeUnknown conditions. Patterns for the TypeClassifier condition are very similar to the patterns for the TypeKnown condition.

The Possible Senses Constraint. We use the learning model of Even-Zohar and Roth (2001), which allows us to constrain the possible senses an input vector can get to the senses of its ASF type. We ran our model without this constraint during both training and test time (recall that for the above results, this constraint was always active during training). In this case, the only difference between the TypeKnown and the TypeUnknown conditions is whether ASF type features are encoded at test time. In the TypeKnown condition, the accuracy of the algorithm drops from 57.9% (when using training and test time constraints and ASF type features) to 53% (when using only ASF type features but no constraints). In the TypeUnknown condition, accuracy drops from 57.24% (when using training time constraints) to 48.03% (when neither constraints nor ASF type features are used). Note

[10] Recall that the performance of the rule-based ASF type classifier on test data is not 100% but 91.6% (Section 6).

[11] $accuracy(algorithm) - accuracy(MFS)$.

that the difference between the constrained model and the unconstrained model is quite large.

The MFS baseline achieves on development data 42.9% and 13.2% in the TypeKnown and TypeUnknown conditions respectively[12]. Thus, the algorithm outperforms the baseline both when the constrained model is used and when an unconstrained multi-class classifier is used.

Note also that when constraints on the possible labels are available at training time, test time constraints and ASF type features (whose inclusion is the difference between the TypeKnown and TypeUnknown) have a minor effect on the results (57.9% for TypeKnown compared to 57.24% for TypeUnknown). However, when training time constraints on the possible labels are not available at training time, ASF type features alone do have a significant effect on the result (53% for TypeKnown compared to 48.03% for TypeUnknown).

POS Features. We next explore the impact of the POS features on the results. These features encode the inflection of the verbs in the CSF, as well as the POS tags of the two words to the left and right of the CSF.

Verb forms provide some partial information corresponding to the ASF type features encoded at the TypeKnown scenario. Table 4 shows that when both label constraints and ASF type features are used, POS features have almost no impact on the final results. When the constrained model is used but ASF type features are not encoded, POS features have an effect on the results. We conclude that when using the constrained model, POS features are important mainly for ASF type information. When the unconstrained classifier is used, POS features have an effect on performance whether ASF type features are encoded or not. In the last case the impact of POS features is larger. In other words, when using an unconstrained classifier, POS features give more than ASF type information to to the model.

Lexical Features. To explore the impact of the lexical features, we removed the following features: time words, reported speech words and 'be' indication features. We saw no impact on model performance when using the constrained model, and a 0.5% decrease when using the unconstrained classifier. That is, our model does not require these lexical features, which is somewhat counter-intuitive. Lexical statistics may turn out to be helpful when using a much larger training set.

Conditional and Wish Features. The conditionals and 'wish' features have a more substantial impact on the results, as they have a role in defining the overall syntactic structure of the sentence. Discarding these features leads to 4% and 1.4% degradation in model accuracy when using the constrained and unconstrained models respectively.

8 Relevant Previous Work

As far as we know, this is the first paper to address the TSD task. In this section we describe related research directions and compare them with TSD.

A relevant task to TSD is WSD (Section 1 and Section 3). Many algorithmic approaches and techniques have been applied to supervised WSD (for reviews see (Agirre and Edmonds, 2006; Mihalcea and Pedersen, 2005; Navigli, 2009)). Among these are various classifiers, ensemble methods combining several supervised classifiers, bootstrapping and semi-supervised learning methods, using the Web as a corpus and knowledge-based methods relying mainly on machine readable dictionaries. Specifically related to this paper are works that exploit syntax (Martinez et al., 2002; Tanaka et al., 2007) and ensemble methods (e.g. (Brody et al., 2006)) to WSD. The references above also describe some unsupervised word sense induction algorithms.

Our TSD algorithm uses the SNOW algorithm, which is a sparse network of classifiers (Section 6). Thus, it most resembles the ensemble approach to WSD. That approach has achieved very good results in several WSD shared tasks (Pedersen, 2000; Florian and Yarowsky, 2002).

Since temporal reasoning is a direct application of TSD, research on this direction is relevant. Such research goes back to (Passonneau, 1988), which introduced the PUNDIT temporal reasoning system. For each tensed clause, PUNDIT first decides whether it refers to an actual time (as in 'We flew TWA to Boston') or not (as in 'Tourists flew TWA to Boston', or 'John always flew his own plane to Boston'). The temporal structure of actual time

[12] Note that these numbers are for development data only.

clauses is then further analyzed. PUNDIT's classification is much simpler than in the TSD task, addressing only actual vs. non-actual time. PUNDIT's algorithmic approach is that of a Prolog rule based system, compared to our statistical learning corpus-based approach. We are not aware of further research that followed their sense disambiguation direction.

Current temporal reasoning research focuses on temporal ordering of events (e.g., (Lapata, 2006; Chambers and Jurafsky, 2008)), for which an accepted atomic task is the identification of the temporal relation between two expressions (see e.g., the TempEval task in SemEval '07 (Verhagen et al., 2007)). This direction is very different from TSD, which deals with the semantics of *individual* concrete tense syntactic forms. In this sense, TSD is an even more atomic task for temporal reasoning.

A potential application of TSD is machine translation where it can assist in translating tense and aspect. Indeed several papers have explored tense and aspect in the MT context. Dorr (1992) explored the integration of tense and aspect information with lexical semantics for machine translation. Schiehlen (2000) analyzed the effect tense understanding has on MT. Ye and Zhang (2005) explored tense tagging in a cross-lingual context. Ye et al., (2006) extracted features for tense translation between Chinese and English. Murata et al., (2007) compared the performance of several MT systems in translating tense and aspect and found that various ML techniques perform better on the task.

Another related field is 'deep' parsing, where a sentence is annotated with a structure containing information that might be relevant for semantic interpretation (e.g. (Hajic, 1998; Baldwin et al., 2007)). TSD senses, however, are not explicitly represented in these grammatical structures, and we are not aware of any work that utilized them to do something close to TSD. This is a good subject for future research.

9 Conclusion and Future Work

In this paper we introduced the Tense Sense Disambiguation (TSD) task, defined as selecting the correct sense of a concrete tense syntactic form in a sentence among the senses of abstract syntactic forms in a syntactic sense dictionary. Unlike in other semantic disambiguation tasks, the sense to be disambiguated is not lexical but of a *syntactic* structure. We prepared a syntactic sense dictionary, annotated a corpus by it, and developed a supervised classifier for sense disambiguation that outperformed a strong baseline.

An obvious direction for future work is to expand the annotated corpus and improve the algorithm by experimenting with additional features. For example, we saw that seeing the full paragraph containing a sentence helps human annotators decide on the appropriate sense which implies that using larger contexts may improve the algorithm.

TSD can be a very useful operation for various high-level applications, for example textual inference, question answering, and information retrieval, in the same way that textual entailment (Dagan et al., 2006) was designed to be. In fact, TSD can assist textual entailment as well, since the sense of a tense form may provide substantial information about the relations entailed from the sentence. Using TSD in such applications is a major direction for future work.

References

Eneko Agirre and Philip Edmonds (Eds). 2006. *Word Sense Disambiguation: Algorithms and Applications*. Springer Verlag.

Timothy Baldwin, Mark Dras, Julia Hockenmaier, Tracy Holloway King, and Gertjan van Noord. 2007. The Impact of Deep Linguistic Processing on Parsing Technology. IWPT '07.

Douglas Biber, Stig Johansson, Geoffrey Leech, Susan Conard, Edward Finegan. 1999. *Longman Grammar of Spoken and Written English*. Longman.

Samuel Brody, Roberto Navigli and Mirella Lapata. 2006. Ensemble Methods for Unsupervised WSD. ACL-COLING '06.

Lou Burnard. 2000. *The British National Corpus User Reference Guide*. Technical Report, Oxford University.

Nathanael Chambers and Dan Jurafsky. 2008. Jointly Combining Implicit Constraints Improves Temporal Ordering. EMNLP '08.

Michelle Cutrer. 1994. Time and Tense in Narratives and in Everyday Language. *PhD dissertation*, University of California at San Diego.

Ido Dagan, Oren Glickman and Bernardo Magnini. 2006. The PASCAL Recognising Textual Entailment Challenge. *Lecture Notes in Computer Science* 2006, 3944:177-190.

John Dinsmore. 1991. *Partitioned representations*. Dordrecht, Netherlands: Kluwer.

Bonnie Dorr. 1992. A Two-Level Knowledge Representation for Machine Translation: Lexical Semantics and Tense/Aspect. In James Pustejovsky and Sabine Bergler, editors, Lexical Semantics and Knowledge Representation.

Yair Even-Zohar and Dan Roth. 2001. A Sequential Model for Multi-Class Classification. EMNLP '01.

Gilles Fauconnier. 2007. *Mental Spaces*. in Dirk Geeraerts and Hubert Cuyckens, editors, The Oxford Handbook of Cognitive Linguistics.

Radu Florian and David Yarowsky. 2002. Modeling Consensus: Classifier Combination for Word Sense Disambiguation. EMNLP '02.

Adele E. Goldberg. 1995. *Constructions: A Construction Grammar Approach to Argument Structure*. University of Chicago Press.

Jan Hajic. 1998. Building a Syntactically Annotated Corpus: The Prague Dependency Treebank. *Issues of Valency and Meaning*, 106–132.

Martin Hewings. 2005. *Advanced Grammar in Use, Second Edition*. Cambridge University University.

Mirella Lapata and Alex Lascarides. 2006. Learning Sentence-internal Temporal Relations. *Journal of Artificial Intelligence Research*, 27:85–117.

Nick Littlestone. 1988. Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm. *Machine Learning*, 285–318.

David MacKay. 2002. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press.

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.

David Martinez, Eneko Agirre, Lluis Marquez. 2002. Syntactic Features for High Precision Word Sense Disambiguation. COLING '02.

Rada Mihalcea and Ted Pedersen. 2005. Advances in Word Sense Disambiguation. Tutorial in ACL '05.

Raymond J. Mooney. 1996. Comparative Experiments on Disambiguating Word Senses: An Illustration of the Role of Bias in Machine Learning. EMNLP '96.

Masaki Murata, Qing Ma, Kiyotaka Uchimoto, Toshiyuki Kanamaru and Hitoshi Isahara. 2007. Japanese-to-English translations of Tense, Aspect, and Modality Using Machine-Learning Methods and Comparison with Cachine-Translation Systems on Market. LREC '07.

Raymond Murphy. 1994. *English Grammar In Use, Second Edition*. Cambridge University Press.

Raymond Murphy. 2007. *Essential Grammar In Use, Third Edition*. Cambridge University Press.

Roberto Navigli. 2009. Word Sense Disambiguation: a Survey. *ACM Computing Surveys*, 41(2) 1–69.

Rebecca J. Passonneau. 1988. A Computational Model of Semantics of Tenses and Aspect. *Computational Linguistics*, 14(2):44–60.

Ted Pedersen. 2000. A Simple Approach to Building Ensembles of Naive Bayesian Classifiers for Word Sense Disambiguation. NAACL '00.

Adwait Ratnaparkhi. 1996. A Maximum Entropy Part-Of-Speech Tagger. EMNLP '06.

Dan Roth. 1998. Learning to Resolve Natural Language Ambiguities: A Unified Approach. AAAI '98.

Michael Schiehlen. 2000. Granularity Effects in Tense Translation. COLING '00.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. SemEval-2007 Task 15: TempEval Temporal Relation Identification. ACL '07.

Takaaki Tanaka, Francis Bond, Timothy Baldwin, Sanae Fujita and Chikara Hashimoto. 2007. Word Sense Disambiguation Incorporating Lexical and Structural Semantic Information. EMNLP-CoNLL '07.

Dave Willis and Jon Wright. 2003. *Collins Cobuild Elementary English Grammar, Second Edition*. HarperCollins Publishers.

Dave Willis. 2004. *Collins Cobuild Intermediate English Grammar, Second Edition*. HarperCollins Publishers.

Yang Ye, Victoria Li Fossum and Steven Abney. 2006. Latent Features in Automatic Tense Translation between Chinese and English. SIGHAN '06.

Yang Ye and Zhu Zhang. 2005. Tense Tagging for Verbs in Cross-Lingual Context: A Case Study. IJCNLP '05.

Harry Zhang. 2004. The Optimality of Naive Bayes. FLAIRS '04.

Improving Mention Detection Robustness to Noisy Input

Radu Florian, John F. Pitrelli, Salim Roukos and Imed Zitouni
IBM T.J. Watson Research Center
Yorktown Heights, NY, U.S.A.
{raduf,pitrelli,roukos,izitouni}us.ibm.com

Abstract

Information-extraction (IE) research typically focuses on clean-text inputs. However, an IE engine serving real applications yields many false alarms due to less-well-formed input. For example, IE in a multilingual broadcast processing system has to deal with inaccurate automatic transcription and translation. The resulting presence of non-target-language text in this case, and non-language material interspersed in data from other applications, raise the research problem of making IE robust to such noisy input text. We address one such IE task: entity-mention detection. We describe augmenting a statistical mention-detection system in order to reduce false alarms from spurious passages. The diverse nature of input noise leads us to pursue a multi-faceted approach to robustness. For our English-language system, at various miss rates we eliminate 97% of false alarms on inputs from other Latin-alphabet languages. In another experiment, representing scenarios in which genre-specific training is infeasible, we process real financial-transactions text containing mixed languages and data-set codes. On these data, because we do not train on data like it, we achieve a smaller but significant improvement. These gains come with virtually no loss in accuracy on clean English text.

1 Introduction

Information-extraction (IE) research is typically performed on clean text in a predetermined language. Lately, IE has improved to the point of being usable for some real-world tasks whose accuracy requirements are reachable with current technology. These uses include media monitoring, topic alerts, summarization, population of databases for advanced search, etc. These uses often combine IE with technologies such as speech recognition, machine translation, topic clustering, and information retrieval.

The propagation of IE technology from isolated use to aggregates with such other technologies, from NLP experts to other types of computer scientists, and from researchers to users, feeds back to the IE research community the need for additional investigation which we loosely refer to as "information-extraction robustness" research. For example:

1. Broadcast monitoring demands that IE handle as input not only clean text, but also the transcripts output by speech recognizers.

2. Multilingual applications, and the imperfection of translation technology, require IE to contend with non-target-language text input (Pitrelli et al., 2008).

3. Naive users at times input to IE other material which deviates from clean text, such as a PDF file that "looks" like plain text.

4. Search applications require IE to deal with databases which not only possess clean text but at times exhibit other complications like markup codes particular to narrow, application-specific data-format standards, for example, the excerpt from a financial-transactions data set shown in Figure 1.

   Legacy industry-specific standards, such as illustrated in this example, are part of long-established processes which are cumbersome to convert to a more-modern database format. Transaction data sets typically build up over a period of years, and as seen here, can exhibit

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 335–345,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

```
:54D://121000358
BANK OF BOSTON
:55D:/0148280005
NEVADA DEPT.OF VET.94C RECOV.FD
-5:MAC:E19DECA8CHK:641EB09B8968

USING OF FIELD 59: ONLY /INS/ WHEN
FOLLOWED BY BCC CODE IN CASE
OF QUESTIONS DONT HESITATE TO
CONTACT US QUOTING REFERENCE
NON-STC CHARGES OR VIA E-MAIL:
YOVANKA(UL)BRATASOVA(AT)BOA.CZ.
BEST REGARDS
BANKA OBCHODNIKA, A.S. PRAGUE, CZ

:58E::ADTX//++ ADDITIONAL
INFORMATION ++ PLEASE BE
INFORMED THAT AS A RESULT OF
THE PURCHASE OFFER ENDED ON 23
MAR 2008 CALDRADE LTD. IS
POSSESSING WITH MORE THEN 90
PER CENT VOTING RIGHT OF SLICE.
THEREFOR CALDRADE LTD. IS
EXERCISING PURCHASE RIGHTS
FOR ALL SLICE SHARES WHICH ARE
CURRENTLY NOT INHIS OWN.
PURCHASE PRICE: HUF 1.940 PER
SHARE. PLEASE :58E::ADTX//NOTE
THAT THOSE SHARES WHICH WILL
NOT BE PRESENTED TO THE OFFER
WILL BE CANCELLED AND INVALID.

:58:SIE SELBST
TRN/REF:515220 035
:78:RUECKGABE DES BETRAGES LT.
ANZBA43 M ZWECKS RUECKGABE IN
AUD. URSPR. ZU UNSEREM ZA MIT
REF. 0170252313279065 UND IHRE
RUECKG. :42:/BNF/UNSERE REF:
```

Figure 1: Example application-specific text, in this case from financial transactions.

peculiar mark-up interspersed with meaningful text. They also suffer complications arising from limited-size entry fields and a diversity of data-entry personnel, leading to effects like haphazard abbreviation and improper spacing, as shown. These issues greatly complicate the IE problem, particularly considering that adapting IE to such formats is hampered by the existence of a multitude of such “standards” and by lack of sufficient annotated data in each one.

A typical state-of-the-art statistical IE engine will happily process such “noisy” inputs, and will typically provide garbage-in/garbage-out performance, embarrassingly reporting spurious “information” no human would ever mistake. Yet it is also inappropriate to discard such documents wholesale: even poor-quality inputs may have relevant information interspersed. This information can include accurate speech-recognition output, names which are recognizable even in wrong-language material, and clean target-language passages interleaved with the mark-up. Thus, here we address methods to make IE robust to such varied-quality inputs. Specifically, our overall goals are

- to skip processing non-language material such as standard or database-specific mark-up,
- to process all non-target-language text cautiously, catching interspersed target-language text as well as text which is compatible with the target language, *e.g.* person names which are the same in the target- and non-target language, and
- to degrade gracefully when processing anomalous target-language material,

while minimizing any disruption of the processing of clean, target-language text, and avoiding any necessity for explicit pre-classification of the genre of material being input to the system. Such explicit classification would be impractical in the presence of the interleaving and the unconstrained data formats from unpredetermined sources.

We begin our robustness work by addressing an important and basic IE task: mention detection (MD). MD is the task of identifying and classifying textual references to entities in open-domain texts. Mentions may be of type “named” (*e.g.* `John`, `Las Vegas`), “nominal” (*e.g.* `engineer`, `dentist`) or “pronominal” (*e.g.* `they`, `he`). A mention also has a specific class which describes the type of entity it refers to. For instance, consider the following sentence:

```
Julia Gillard, prime
minister of Australia,
declared she will enhance
the country's economy.
```

Here we see three mentions of one person entity: `Julia Gillard`, `prime minister`, and `she`; these mentions are of type named, nominal, and pronominal, respectively. `Australia` and `country` are mentions of type named and nominal, respectively, of a single geopolitical entity. Thus, the MD task is a more general and complex task than named-entity recognition, which aims at identifying and classifying only named mentions.

Our approach to IE has been to use language-independent algorithms, in order to facilitate reuse across languages, but we train them with language-specific data, for the sake of accuracy. Therefore, input is expected to be predominantly in a target language. However, real-world data genres inevitably include some mixed-language/non-linguistic input. Genre-specific training is typically infeasible due to such application-specific data sets being unannotated, motivating this line of research. Therefore, the goal of this study is to investigate schemes to make a language-specific MD engine robust to the types of interspersed non-target material described above. In these initial experiments, we work with English as the target language, though we aim to make our approach to robustness as target-language-independent as possible.

While our ultimate goal is a language-independent approach to robustness, in these initial experiments, English is the target language. However, we process mixed-language material including real-world data with its own peculiar mark-up, text conventions including abbreviations, and mix of languages, with the goal of English MD.

We approach robust MD using a multi-stage strategy. First, non-target-character-set passages (here, non-Latin-alphabet) are identified and marked for non-processing. Then, following word-tokenization, we apply a language classifier to a sliding variable-length set of windows in order to generate features for each word indicative of how much the text around that word resembles good English, primarily in comparison to other Latin-alphabet languages. These features are used in a separate maximum-entropy classifier whose output is a single feature to add to the MD classifier. Additional features, primarily to distinguish English from non-language input, are added to MD as well. An example is the minimum of the number of letters and the number of digits in the "word", which when greater than zero often indicates database detritus. Then we run the MD classifier enhanced with these new robustness-oriented features. We evaluate using a detection-error-trade-off (DET) (Martin et al., 1997) analysis, in addition to traditional precision/recall/F-measure.

This paper is organized as follows. Section 2 discusses previous work. Section 3 describes the baseline maximum-entropy-based MD system. Section 4 introduces enhancements to the system to achieve robustness. Section 5 describes databases used for experiments, which are discussed in Section 6, and Section 7 draws conclusions and plots future work.

2 Previous work on mention detection

The MD task has close ties to named-entity recognition, which has been the focus of much recent research (Bikel et al., 1997; Borthwick et al., 1998; Tjong Kim Sang, 2002; Florian et al., 2003; Benajiba et al., 2009), and has been at the center of several evaluations: MUC-6, MUC-7, CoNLL'02 and CoNLL'03 shared tasks. Usually, in computational-linguistics literature, a named entity represents an instance of either a location, a person, an organization, and the named-entity-recognition task consists of identifying each individual occurrence of names of such an entity appearing in the text. As stated earlier, in this paper we are interested in identification and classification of textual references to object/abstraction *mentions*, which can be either named, nominal or pronominal. This task has been a focus of interest in ACE since 2003. The recent ACE evaluation campaign was in 2008.

Effort to handle noisy data is still limited, especially for scenarios in which the system at decoding time does not have prior knowledge of the input data source. Previous work dealing with unstructured data assumes the knowledge of the input data source. As an example, E. Minkov *et al.* (Minkov et al., 2005) assume that the input data is text from e-mails, and define special features to enhance the detection of named entities. Miller *et al.* (Miller et al., 2000) assume that the input data is the output of a speech or optical character recognition system, and hence extract new features for better named-entity recognition. In a different research problem, L. Yi *et al.* eliminate the noisy text from the document before performing data mining (Yi et al., 2003). Hence, they do not try to process noisy data; instead, they remove it. The approach we propose in this paper does not assume prior knowledge of the data source. Also we do not want to eliminate the noisy data, but rather attempt to detect the appropriate mentions, if any, that appear in that portion of the data.

3 Mention-detection algorithm

Similarly to classical NLP tasks such as base phrase chunking (Ramshaw and Marcus, 1999) and named-entity recognition (Tjong Kim Sang, 2002), we formulate the MD task as a sequence-classification problem, by assigning to each word token in the text a label indicating whether it starts a specific mention, is inside a specific mention, or is outside any mentions. We also assign to every non-outside label a class to specify entity type *e.g.* person, organization, location, etc. We are interested in a statistical approach that can easily be adapted for several languages and that has the ability to integrate easily and make effective use of diverse sources of information to achieve high system performance. This is because, similar to many NLP tasks, good performance has been shown to depend heavily on integrating many sources of information (Florian et al., 2004). We choose a Maximum Entropy Markov Model (MEMM) as described previously (Florian et al., 2004; Zitouni and Florian, 2009). The maximum-entropy model is trained using the *sequential conditional generalized iterative scaling* (SCGIS) technique (Goodman, 2002), and it uses a *Gaussian prior* for regularization (Chen and Rosenfeld, 2000)[1].

3.1 Mention detection: standard features

The featues used by our mention detection systems can be divided into the following categories:

1. **Lexical Features** Lexical features are implemented as token n-grams spanning the current token, both preceding and following it. For a token x_i, token n-gram features will contain the previous $n-1$ tokens ($x_{i-n+1}, \dots x_{i-1}$) and the following $n-1$ tokens ($x_{i+1}, \dots x_{i+n-1}$). Setting n equal to 3 turned out to be a good choice.

2. **Gazetteer-based Features** The gazetteer-based features we use are computed on tokens. The gazetteers consist of several class of dictionaries: including person names, country names, company names, etc. Dictionaries contain single names such as `John` or `Boston`, and also phrases such as `Barack Obama`, `New York City`, or `The United States`. During both training and decoding, when we encounter in the text a token or a sequence of tokens that completely matches an entry in a dictionary, we fire its corresponding class.

The use of this framework to build MD systems for clean English text has given very competitive results at ACE evaluations (Florian et al., 2006). Trying other classifiers is always a good experiment, which we didn't pursue here for two reasons: first, the MEMM system used here is state-of-the-art, as proven in evaluations and competitions – while it is entirely possible that another system might get better results, we don't think the difference would be large. Second, we are interested in ways of improving performance on noisy data, and we expect any system to observe similar degradation in performance when presented with unexpected input – showing results for multiple classifier types might very well dilute the message, so we stuck to one classifier type.

4 Enhancements for robustness

As stated above, our goal is to skip spans of characters which do not lend themselves to target-language MD, while minimizing impact on MD for target-language text, with English as the initial target language for our experiments. More specifically, our task is to process data automatically in any unpredetermined format from any source, during which we strive to avoid outputting spurious mentions on:

- non-language material, such as mark-up tags and other data-set detritus, as well as non-text data such as code or binaries likely mistakenly submitted to the MD system,
- non-target-character-set material, here, non-Latin-alphabet material, such as Arabic and Chinese in their native character sets, and
- target-character-set material not in the target language, here, Latin-alphabet languages other than English.

It is important to note that this is not merely a document-classification problem; this non-target data is often interspersed with valid input text.

[1] Note that the resulting model cannot really be called a maximum-entropy model, as it does not yield the model which has the maximum entropy (the second term in the product), but rather is a maximum-*a-posteriori* model.

Mark-up is the obvious example of interspersing; however, other categories of non-target data can also interleave tightly with valid input. A few examples:

- English text is sometimes infixed right in a Chinese sentence, such as 其他BBC网站
- some translation algorithms will leave unchanged an untranslatable word, or will transliterate it into the target language using a character convention which may not be a standard known to the MD engine, and
- some target-alphabet-but-non-target-language material will be compatible with the target language, particularly people's names. An example with English as the target language is `Barack Obama` in the Spanish text `...presidente de Estados Unidos, Barack Obama, dijo el da 24 que ....`

Therefore, to minimize needless loss of processable material, a robustness algorithm ideally does a sliding analysis, in which, character-by-character or word-by-word, material may be deemed to be suitable to process. Furthermore, a variety of strategies will be needed to contend with the diverse nature of non-target material and the patterns in which it will appear among valid input.

Accordingly, the following is a summary of algorithmic enhancements to MD:

1. detection of standard file formats, such as SGML, and associated detagging,
2. segmentation of the file into target- vs. non-target-character-set passages, such that the latter not be processed further,
3. tokenization to determine word and sentence units, and
4. MD, augmented as follows:
   - Sentence-level categorization of likelihood of good English.
   - If "clean" English was detected, run the same clean baseline model as described in Section 3.
   - If the text is determined to be a bad fit to English, run an alternate maximum-entropy model that is heavily based on gazetteers, using only context-independent (*e.g.* primarily gazetteer-based) features, to catch isolated obvious English/English-compatible names embedded in otherwise-foreign text.
   - If in between "clean" and "bad", use a "mixed" maximum-entropy MD model whose training data and feature set are augmented to handle interleaving of English with mark-up and other languages.

These MD-algorithm enhancements will be described in the following subsections.

4.1 Detection and detagging for standard file formats

Some types of mark-up are well-known standards, such as SGML (Warmer and van Egmond, 1989). Clearly the optimal way of dealing with them is to apply detectors of these specific formats, and associated detaggers, as done previously (Yi et al., 2003). For this reason, standard mark-up is not a subject of the current study; rather, our concern is with mark-up peculiar to specific data sets, as described above, and so while this step is part of our overall strategy, it is not employed in the present experiments.

4.2 Character-set segmentation

Some entity mentions may be recognizable in a non-target language which shares the target-language's character set, for example, a person's name recognizable by English speakers in an otherwise-not-understandable Spanish sentence. However, non-target character sets, such as Arabic and Chinese when processing English, represent pure noise for an IE system. Therefore, deterministic character-set segmentation is applied, to mark non-target-character-set passages for non-processing by the remainder of the system, or, in a multilingual system, to be diverted to a subsystem suited to process that character set. Characters which can be ambiguous with regard to character set, such as some punctuation marks, are attached to target-character-set passages when possible, but are not considered to break non-target-character-set passages surrounding them on both sides.

4.3 Tokenization

Subsequent processing is based on determination of the language of target-alphabet text. The fundamental unit of such processing is target-alphabet word, necessitating tokenization at this point into word-level units. This step includes punctuation separation as well as the detction of sentence boundary (Zimmerman et al., 2006).

4.4 Robust mention detection

After preprocessing steps presented earlier, we detect mentions using a cascaded approach that combines several MD classifiers. Our goal is to select among maximum-entropy MD classifiers trained separately to represent different degrees of "noisiness" occurring in many genres of data, including machine-translation output, informal communications, mixed-language material, varied forms of non-standard database mark-up, etc. We somewhat-arbitrarily choose to employ three classifiers as described below. We select a classifier based on a sentence-level determination of the material's fit to the target language. First, we build an n-gram language model on clean target-language training text. This language model is used to compute the perplexity (PP) of each sentence during decoding. The PP indicates the quality of the text in the target-language (*i.e.* English) (Brown et al., 1992); the lower the PP, the cleaner the text. A sentence with a PP lower than a threshold θ_1 is considered "clean" and hence the "clean" baseline MD model described in Section 3 is used to detect mentions of this sentence. The clean MD model has access to standard features described in Section 3.1. In the case where a sentence looks particularly badly matched to the target language, defined as $PP > \theta_2$, we use a "*gazetteer-based*" model based on a dictionary look-up to detect mentions; we retreat to seeking known mentions in a context-independent manner reflecting that most of the context consists of out-of-vocabulary words. The gazetteer-based MD model has access only to gazetteer information and does not look to lexical context during decoding, reflecting the likelihood that in this poor material, words surrounding any recognizable mention are foreign and therefore unusable. In the case of an in-between determination, that is, a sentence with $\theta_1 < PP < \theta_2$, we use a "*mixed*" MD model, based on augmenting the training data set and the feature set as described in the next section. The values of θ_1 and θ_2 are estimated empirically on a separate development data set that is also used to tune the Gaussian prior (Chen and Rosenfeld, 2000). This set contains a mix of clean English and Latin-alphabet-but-non-English text that is not used for traning and evaluation.

The advantage of this combination strategy is that we do not need pre-defined knowledge of the text source in order to apply an appropriate model. The selection of the appropriate model to use for decoding is done automatically based on PP value of the sentence. We will show in the experiments section how this combination strategy is effective not only in maintaining good performance on a clean English text but also in improving performance on non-English data when compared to other source-specific MD models.

4.5 Mixed mention detection model

The mixed MD model is designed to process "sentences" mixing English with non-English, whether foreign-language or non-language material. Our approach is to augment model training compared to the clean baseline by adding non-English, mixed-language, and non-language material, and to augment the model's feature set with language-identification features more localized than the sentence-level perplexity described above, as well as other features designed primarily to distinguish non-language material such as mark-up codes.

4.5.1 Language-identification features

We apply an n-gram-based language classifier (Prager, 1999) to variable-length sliding windows as follows. For each word, we run 1- through 6-preceding-word windows through the classifier, and 1- through 6-word windows beginning with the word, for a total of 12 windows, yielding for each window a result like:

```
0.235 Swedish
0.148 English
0.134 French
...
```

For each of the 12 results, we extract three features: the identity of the top-scoring language, here, Swedish; the confidence score in the top-scoring language, here, 0.235; and the score difference between the target language (English for these experiments) and the top-scoring non-target language, here, $0.148 - 0.235 = -0.087$. Thus we have a 36-feature vector for each word. We bin these and use them as input to a maximum-entropy classifier (separate from the MD classifier) which outputs "English" or "Non-English", and a confidence score. These scores in turn are binned into six categories to serve as a "how-English-is-it" feature in the augmented MD model. The language-identification classifier and the maximum-entropy "how-English" classifier are each trained on text data separate from each other and from the training and test sets for MD.

4.5.2 Additional features

The following features are designed to capture evidence of whether a "word" is in fact linguistic material or not: number of alphabetic characters, number of characters, maximum consecutive repetitions of a character, numbers of non-alphabetic and non-alphanumeric characters, fraction of characters which are alphabetic, fraction alphanumeric, and number of vowels. These features are part of the augmentation of the mixed MD model relative to the clean MD model.

5 Data sets

Four data sets are used for our initial experiments. One, "English", consists of 367 documents totaling 170,000 words, drawn from web news stories from various sources and detagged to be plain text. This set is divided into 340 documents as a training set and 27 for testing, annotated as described in more detail elsewhere (Han, 2010). These data average approximately 21 annotated mentions per 100 words.

The second set, "Latin", consists of 23 detagged web news articles from 11 non-English Latin-alphabet languages totaling 31,000 words. Of these articles, 12 articles containing 19,000 words are used as a training set, with the remaining used for testing, and each set containing all 11 languages. They are annotated using the same annotation conventions as "English", and from the perspective of English; that is, only mentions which would be clear to an English speaker are labeled, such as `Barack Obama` in the Spanish example in Section 4. For this reason, these data average only approximately 5 mentions per 100 words.

The third, "Transactions", consists of approximately 60,000 words drawn from a text data set logging real financial transactions. Figure 1 shows example passages from this database, anonymized while preserving the character of the content.

This data set logs transactions by a staff of customer-service representatives. English is the primary language, but owing to international clientele, occasionally representatives communicate in other languages, such as the German here, or in English but mentioning institutions in other countries, here, a Czech bank. Interspersed among text are codes specific to this application which delineate and identify various information fields and punctuate long passages. The application also places constraints on legal characters, leading to the unusual representation of underline and the "at" sign as shown, making for an e-mail address which is human-readable but likely not obvious to a machine. Abbreviations represent terms particularly common in this application area, though they may not be obvious without adapting to the application; these include standards like `HUF`, a currency code which stands for Hungarian forint, and financial-transaction peculiarities like `BNF` for "beneficiary" as seen in Figure 1. In short, good English is interspersed with non-language content, foreign-language text, and rough English like data-entry errors and haphazard abbreviations. These data average 4 mentions per 100 words.

Data sets with peculiarities analogous to those in this Transactions set are commonplace in a variety of settings. Training specific to data sets like this is often infeasible due to lack of labeled data, insufficient data for training, and the multitude of such data formats. For this reason, we do not train on Transactions, letting our testing on this data set serve as an example of testing on such data formats unseen.

6 Experiments

MD systems were trained to recognize the 116 entity-mention types shown in Table 1, annotated as described previously (Han, 2010). The clean-data classifier was trained on the English training data using the feature set described in Section 3.1. The classifier for "mixed"-quality data and the "gazetteer" model were each trained on that set plus the "Latin" training set and the supplemental set. In addition, "mixed" training included the additional features described in Section 4.5. The framework used to build the baseline MD system is similar to the one we used in the ACE evaluation[2]. This system has achieved competitive results with an F-measure of 82.7 when trained on the seven main types of ACE data with access to wordnet and part-of-speech-tag information as well as output of other MD and named-entity recognizers (Zitouni and Florian, 2008).

It is instructive to evaluate on the individual component systems as well as the combination, despite the fact that the individual components are not well-suited to all the data sets, for example, the mixed and gazetteer systems being a poorer fit to the English task than the baseline, and vice versa for the

[2] NIST's ACE evaluation plan: http://www.nist.gov/speech/tests/ace/index.htm

age	event-custody	facility	people	date
animal	event-demonstration	food	percent	duration
award	event-disaster	geological-object	person	e-mail-address
cardinal	event-legal	geo-political	product	measure
disease	event-meeting	law	substance	money
event	event-performance	location	title-of-a-work	phone-number
event-award	event-personnel	ordinal	vehicle	ticker-symbol
event-communication	event-sports	organ	weapon	time
event-crime	event-violence	organization	web-address	

Table 1: Entity-type categories used in these experiments. The eight in the right-most column are not further distinguished by mention type, while the remaining 36 are further classified as named, nominal or pronominal, for a total of $36 \times 3 + 8 = 116$ mention labels.

	English			**Latin**			**Transactions**		
	P	R	F	P	R	F	P	R	F
Clean	78.7	73.6	76.1	16.0	40.0	22.9	19.5	32.2	24.3
Mixed	77.9	69.7	73.6	78.5	55.9	65.3	37.1	47.8	41.7
Gazetteer	76.9	66.2	71.1	77.8	55.5	64.8	36.5	47.5	41.3
Combination	78.1	73.2	**75.6**	80.4	56.0	**66.0**	38.5	49.1	**43.2**

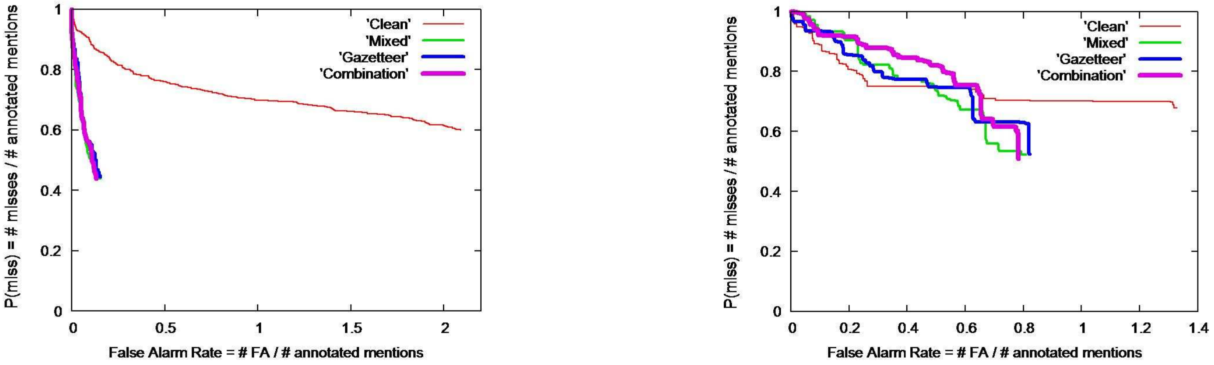
Table 2: Performance of clean, mixed, and gazetteer-based mention detection systems as well as their combination. Performance is presented in terms of Precision (P), Recall (R), and F-measure (F).

non-target data sets. Precision/recall/F-measure results are shown in Table 2. Not surprisingly, the baseline system, intended for clean data, performs poorly on noisy data. The mixed and gazetteer systems, having a variety of noisy data in their training set, perform much better on the noisy conditions, particularly on Latin-alphabet-non-English data because that is one of the conditions included in its training, while Transactions remains a condition not covered in the training set and so shows less improvement. However, because the mixed classifier, and moreso the gazetteer classifier, are oriented to noisy data, on clean data they suffer in performance by 2.5 and 5 F-measure points, respectively. But system combination serves us well: it recovers all but 0.5 F-measure point of this loss, while also actually performing better on the noisy data sets than the two classifiers specifically targeted toward them, as can be seen in Table 2. It is important to note that the major advantage of using the combination model is the fact that we do not have to know the data source in order to select the appropriate MD model to use. We assume that the data source is unknown, which is our claim in this work, and we show that we obtain better performance than using source-specific MD models. This reflects the fact that a noisy data set will in fact have portions with varying degrees of “noise”, so the combination outperforms any single model targeted to a single particular level of noise, enabling the system to contend with such variability without the need for pre-segregating sub-types of data for noise level. The obtained improvement from the system combination over all other models is statistically significant based on the stratified bootstrap re-sampling significance test (Noreen, 1989). We consider results statistically significant when $p < 0.05$, which is the case in this paper. This approach was used in the named-entity-recognition shared task of CoNNL-2002[3].

It should be noted that some completely-non-target types of data, such as non-target-character set data, have been omitted from analysis here. Including them would make our system look comparatively stronger, as they would have only spurious mentions and so generate false alarms but no correct mentions in the baseline system, while our system deterministically removes them.

As mentioned above, we view MD robustness primarily as an effort to eliminate, relative to a baseline system, large volumes of spurious “mentions” detected in non-target input content, while minimiz-

[3] http://www.cnts.ua.ac.be/conll2002/ner/



(a) DET plot for clean (baseline), mixed, gazetteer, and combination MD systems on the Latin-alphabet-non-English text. The clean system (upper curve) performs far worse than the other three systems designed to provide robustness; these systems in turn perform nearly indistinguishably.

(b) DET plot for clean (baseline), mixed, gazetteer, and combination MD systems on the Transactions data set. The clean system (upper/longer curve) reaches far higher false-alarm rates, while never approaching the lower miss rates achievable by any of the other three systems, which in turn perform comparably to each other.

Figure 2: DET plots for Latin-alphabet-non-English and Transactions data sets

ing disruption of detection in target input. A secondary goal is recall in the event of occasional valid mentions in such non-target material. Thus, as input material degrades, precision increases in importance relative to recall. As such, we view precision and recall asymmetrically on this task, and so rather than evaluating purely in terms of F-measure, we perform a detection-error-trade-off (DET) (Martin et al., 1997) analysis, in which we plot a curve of miss rate on valid mentions vs. false-alarm rate, with the curve traced by varying a confidence threshold across its range. We measure false-alarm and miss rates relative to the number of actual mentions annotated in the data set:

$$\text{FA rate} = \frac{\#\text{ false alarms}}{\#\text{ annotated mentions}} \qquad (1)$$

$$\text{Miss rate} = \frac{\#\text{ misses}}{\#\text{ annotated mentions}} \qquad (2)$$

where false alarms are "mentions" output by the system but not appearing in annotation, while misses are mentions which are annotated but do not appear in the system output. Each mention is treated equally in this analysis, so frequently-recurring entity/mention types weigh on the results accordingly.

Figure 2a shows a DET plot for the clean, mixed, gazetteer, and combination systems on the "Latin" data set, while Figure 2b shows the analogous plot for the "Transactions" data set. The drastic gains made over the baseline system by the three experimental systems are evident in the plots. For example, on Latin, choosing an operating point of a miss rate of 0.6 (nearly the best achievable by the clean system), we find that the robustness-oriented systems eliminate 97% of the false alarms of the clean baseline system, as the plot shows false-alarm rates near 0.07 compared to the baseline's of 2.08. Gains on Transaction data are more modest, owing to this case representing a data genre not included in training. It should be noted that the jaggedness of the Transaction curves traces to the repetitive nature of some of the terms in this data set.

In making a system more oriented toward robustness in the face of non-target inputs, it is important to quantify the effect of these systems being less-oriented toward clean, target-language text. Figure 3 shows the analogous DET plot for the English test set, showing that achieving robustness through the combination system comes at a small cost to accuracy on the text the original system is trained to process.

7 Conclusions

For information-extraction systems to be useful, their performance must degrade gracefully when confronted with inputs which deviate from ideal and/or derive from unknown sources in unknown formats. Imperfectly-translated, mixed-language, marked-up text and non-language material must not

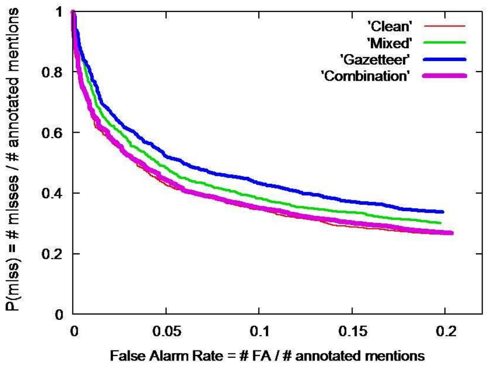


Figure 3: DET plot for clean (baseline), mixed, gazetteer, and combination MD systems on clean English text, verifying that performance by the clean system (lowest curve) is very closely approximated by the combination system (second-lowest curve), while the mixed system performs somewhat worse and the gazetteer system (top curve), worse still, reflecting that these systems are increasingly oriented toward noisy inputs.

be processed in a garbage-in-garbage-out fashion merely because the system was designed only to handle clean text in one language. Thus we have embarked on information-extraction-robustness work, to improve performance on imperfect inputs while minimizing disruption of processing of clean text. We have demonstrated that for one IE task, mention detection, a multi-faceted approach, motivated by the diversity of input data imperfections, can eliminate a large proportion of the spurious outputs compared to a system trained on the target input, at a relatively small cost of accuracy on that target input. This outcome is achieved by a system-combination approach in which a perplexity-based measure of how well the input matches the target language is used to select among models designed to deal with such varying levels of noise. Rather than relying on explicit recognition of genre of source data, the experimental system merely does its own assessment of how much each sentence-sized chunk matches the target language, an important feature in the case of unknown text sources.

Chief among directions for further work is to continue to improve performance on noisy data, and to strengthen our findings via larger data sets. Additionally, we look forward to expanding analysis to different types of imperfect input, such as machine-translation output, different types of mark-up, and different genres of real data. Further work should also explore the degree to which the approach to achieving robustness must vary according to the target language. Finally, robustness work should be expanded to other information-extraction tasks.

Acknowledgements

The authors thank Ben Han, Anuska Renta, Veronique Baloup-Kovalenko and Owais Akhtar for their help with annotation. This work was supported in part by DARPA under contract HR0011-08-C-0110.

References

Y. Benajiba, M. Diab, and P. Rosso. 2009. Arabic named entity recognition: A feature-driven study. *In the special issue on Processing Morphologically Rich Languages of the IEEE Transaction on Audio, Speech and Language*.

D. M. Bikel, S. Miller, R. Schwartz, and R. Weischedel. 1997. Nymble: a high-performance learning name-finder. In *Proceedings of ANLP-97*, pages 194–201.

A. Borthwick, J. Sterling, E. Agichtein, and R. Grishman. 1998. Exploiting diverse knowledge sources via maximum entropy in named entity recognition.

P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, J. C. Lai, and R. L. Mercer. 1992. An estimate of an upper bound for the entropy of English. *Computational Linguistics*, 18(1), March.

S. Chen and R. Rosenfeld. 2000. A survey of smoothing techniques for ME models. *IEEE Transaction on Speech and Audio Processing*.

R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. 2003. Named entity recognition through classifier combination. In *Conference on Computational Natural Language Learning - CoNLL-2003*, Edmonton, Canada, May.

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 *Proceedings of HLT-NAACL 2004*, pages 1–8.

R. Florian, H. Jing, N. Kambhatla, and I. Zitouni. 2006. Factorizing complex models: A case study in mention detection. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 473–480, Sydney, Australia, July. Association for Computational Linguistics.

J. Goodman. 2002. Sequential conditional generalized iterative scaling. In *Proceedings of ACL'02*.

D. B. Han. 2010. Klue annotation guidelines - version 2.0. Technical Report RC25042, IBM Research, August.

A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki. 1997. The DET curve in assessment of detection task performance. In *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, pages 1895–1898. Rhodes, Greece.

D. Miller, S. Boisen, R. Schwartz, R. Stone, and R. Weischedel. 2000. Named entity extraction from noisy input: speech and OCR. In *Proceedings of the sixth conference on Applied natural language processing*, pages 316–324, Morristown, NJ, USA. Association for Computational Linguistics.

E. Minkov, R. C. Wang, and W. W. Cohen. 2005. Extracting personal names from email: Applying named entity recognition to informal text. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 443–450, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

E. W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses*. John Wiley Sons.

J. F. Pitrelli, B. L. Lewis, E. A. Epstein, M. Franz, D. Kiecza, J. L. Quinn, G. Ramaswamy, A. Srivastava, and P. Virga. 2008. Aggregating Distributed STT, MT, and Information Extraction Engines: The GALE Interoperability-Demo System. In *Interspeech*. Brisbane, NSW, Australia.

J. M. Prager. 1999. Linguini: Language identification for multilingual documents. In *Journal of Management Information Systems*, pages 1–11.

L. Ramshaw and M. 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*, pages 157–176. Kluwer.

E. F. Tjong Kim Sang. 2002. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2002*, pages 155–158. Taipei, Taiwan.

J. Warmer and S. van Egmond. 1989. The implementation of the Amsterdam SGML parser. *Electron. Publ. Origin. Dissem. Des.*, 2(2):65–90.

L. Yi, B. Liu, and X. Li. 2003. Eliminating noisy information in web pages for data mining. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 296–305, New York, NY, USA. ACM.

M. Zimmerman, D. Hakkani-Tur, J. Fung, N. Mirghafori, L. Gottlieb, E. Shriberg, and Y. Liu. 2006. The ICSI+ multilingual sentence segmentation system. In *Interspeech*, pages 117–120, Pittsburgh, Pennsylvania, September.

I. Zitouni and R. Florian. 2008. Mention detection crossing the language barrier. In *Proceedings of EMNLP'08*, Honolulu, Hawaii, October.

I. Zitouni and R. Florian. 2009. Cross-language information propagation for Arabic mention detection. *ACM Transactions on Asian Language Information Processing (TALIP)*, 8(4):1–21.

Clustering-based Stratified Seed Sampling for Semi-Supervised Relation Classification

Longhua Qian
Natural Language Processing Lab
School of Computer Science and Technology
Soochow University
1 Shizi Street, Suzhou, China 215006
qianlonghua@suda.edu.cn

Guodong Zhou
Natural Language Processing Lab
School of Computer Science and Technology
Soochow University
1 Shizi Street, Suzhou, China 215006
gdzhou@suda.edu.cn

Abstract

Seed sampling is critical in semi-supervised learning. This paper proposes a clustering-based stratified seed sampling approach to semi-supervised learning. First, various clustering algorithms are explored to partition the unlabeled instances into different strata with each stratum represented by a center. Then, diversity-motivated intra-stratum sampling is adopted to choose the center and additional instances from each stratum to form the unlabeled seed set for an oracle to annotate. Finally, the labeled seed set is fed into a bootstrapping procedure as the initial labeled data. We systematically evaluate our stratified bootstrapping approach in the semantic relation classification subtask of the ACE RDC (Relation Detection and Classification) task. In particular, we compare various clustering algorithms on the stratified bootstrapping performance. Experimental results on the ACE RDC 2004 corpus show that our clustering-based stratified bootstrapping approach achieves the best F1-score of 75.9 on the subtask of semantic relation classification, approaching the one with golden clustering.

1 Introduction

Semantic relation extraction aims to detect and classify semantic relationships between a pair of named entities occurring in a natural language text. Many machine learning approaches have been proposed to attack this problem, including supervised (Miller et al., 2000; Zelenko et al., 2003; Culotta and Soresen, 2004; Kambhatla, 2004; Zhao and Grishman, 2005; Zhou et al., 2005; Zhang et al., 2006; Zhou and Zhang, 2007; Zhou et al., 2007; Qian et al., 2008; Zhou et al., 2010), semi-supervised (Brin, 1998; Agichtein and Gravano, 2000; Zhang, 2004; Chen et al., 2006; Qian et al., 2009; Zhou et al., 2009), and unsupervised methods (Hasegawa et al., 2004; Zhang et al., 2005; Chen et al., 2005).

Current work on relation extraction mainly adopts supervised learning methods, since they achieve much better performance. However, they normally require a large number of manually labeled relation instances, whose acquisition is both time consuming and labor intensive. In contrast, unsupervised methods do not need any manually labeled instances. Nevertheless, it is difficult to assess their performance due to the lack of evaluation criteria. As something between them, semi-supervised learning has received more and more attention recently. With the plenitude of unlabeled natural language text at hand, semi-supervised learning can significantly reduce the need for labeled data with only limited sacrifice in performance. For example, Abney (2002) proposes a bootstrapping algorithm which chooses the unlabeled instances with the highest probability of being correctly labeled and add them in turn into the labeled training data iteratively.

This paper focuses on bootstrapping-based semi-supervised learning in relation extraction. Since the performance of bootstrapping depends much on the quality and quantity of the seed set and researchers tend to employ as few seeds as possible (e.g. 100 instances) to save time and labor, the quality of the seed set plays a critical role in bootstrapping. Furthermore, the imbalance of different classes and

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 346–355,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

the inherent structural complexity of instances will severely weaken the strength of bootstrapping and semi-supervised learning as well. Therefore, it is critical for a bootstrapping procedure to select an appropriate seed set, which should be representative and diverse. However, most of current semi-supervised relation extraction systems (Zhang, 2004; Chen et al., 2006) use a random seed sampling strategy, which fails to fully exploit the affinity nature in the training data to derive the seed set. Alternatively, Zhou et al. (2009) bootstrap a set of weighted support vectors from both labeled and unlabeled data using SVM and feed these instances into semi-supervised relation extraction. However, their seed set is sequentially generated only to ensure that there are at least 5 instances for each relation class. Our previous work (Qian et al., 2009) attempts to solve this problem via a simple stratified sampling strategy for selecting the seed set. Experimentation on the ACE RDC 2004 corpus shows that the stratified sampling strategy achieves promising results for semi-supervised learning. Nevertheless, the success of the strategy relies on the assumption that the true distribution of all relation types is already known, which is impractical for real NLP applications.

This paper presents a clustering-based stratified seed sampling approach for semi-supervised relation extraction, without the assumption on the true distribution of different relation types. The motivations behind our approach are that the unlabeled data can be partitioned into a number of strata using a clustering algorithm and that representative and diverse seeds can be derived from such strata in the framework of stratified sampling (Neyman, 1934) for an oracle to annotate. Particularly, we employ a diversity-motivated intra-stratum sampling scheme to pick a center and additional instances as seeds from each stratum. Experimental results show the effectiveness of the clustering-based stratified seed sampling for semi-supervised relation classification.

The rest of this paper is organized as follows. First an overview of the related work is given in Section 2. Then, Section 3 introduces the stratified bootstrapping framework including an intra-stratum sampling scheme while Section 4 describes various clustering algorithms. The experimental results on the ACE RDC 2004 corpus are reported in Section 5. Finally we conclude our work and indicate some future directions in Section 6.

2 Related Work

In semi-supervised learning for relation extraction, most of previous work construct the seed set either randomly (Zhang, 2004; Chen et al., 2006) or sequentially (Zhou et al., 2009). Qian et al. (2009) adopt a stratified sampling strategy to select the seed set. However, their method needs a stratification variable such as the known distribution of the relation types, while our method uses clustering to divide relation instances into different strata.

In the literature, clustering techniques have been employed in active learning to sample representative seeds in a certain extent (Nguyen and Smeulders, 2004; Tang et al., 2002; Shen et al., 2004). Our work is similar to the formal framework, as proposed in Nguyen and Smeulders (2004), in which K-medoids clustering is incorporated into active learning. The cluster centers are used to construct a classifier and which in turn propagates classification decision to other examples via a local noise model. Unlike their probabilistic models, we apply various clustering algorithms together with intra-stratum sampling to select a seed set in discriminative models like SVMs. In active learning for syntactic parsing, Tang et al. (2002) employ a sampling strategy of “most uncertain per cluster” to select representative examples and weight them using their cluster density, while we pick a few seeds (the number of the sampled seeds is proportional to the cluster density) from a cluster in addition to its center. Shen et al. (2004) combine multiple criteria to measure the informativeness, representativeness, and diversity of examples in active learning for named entity recognition. Unlike our sampling strategy of clustering for representativeness and stratified sampling for diversity, they either select cluster centroids or diverse examples from a prechosen set in terms of some combined metrics. To the best of our knowledge, this is the first work to address the issue of seed selection using clustering techniques for semi-supervised learning with discriminative models.

3 Stratified Bootstrapping Framework

The stratified bootstrapping framework consists of three major components: an underlying supervised learner and a bootstrapping algorithm on top of it as usual, plus a clustering-based stratified seed sampler.

3.1 Underlying Supervised Learner

Due to recent success in tree kernel-based relation extraction, this paper adopts a tree kernel-based method in the underlying supervised learner. Following the previous work in relation extraction (Zhang et al., 2006; Zhou et al., 2007; Qian et al., 2008), we use the standard convolution tree kernel (Collins and Duffy, 2001) to count the number of common sub-trees as the structural similarity between two parse trees. Besides, to properly represent a relation instance, this paper adopts the Unified Parse and Semantic Tree (UPST), as proposed in Qian et al. (2008). To our knowledge, the USPT has achieved the best performance in relation extraction so far on the ACE RDC 2004 corpus.

In particular, we use the SVM^{light}-TK[1] package as our classifier. Since the package is a binary classifier, we adapt it to the multi-class tasks of relation extraction by applying the *one vs. others* strategy, which builds *K* binary classifiers so as to separate one class from all others. The final classification decision of an instance is determined by the class that has the maximal SVM output margin.

3.2 Bootstrapping Algorithm

Following Zhang (2004), we have developed a baseline self-bootstrapping procedure, which keeps augmenting the labeled data by employing the models trained from previously available labeled data, as shown in Figure 1.

Since the SVM^{light}-TK package doesn't output any probability that it assigns to the class label on an instance, we devise a metric to measure the confidence with regard to the classifier's prediction. Given a sequence of output margins of all *K* binary classifiers at some iteration, denoted as $\{m_1, m_2, \ldots m_K\}$ with m_i the margin for the *i*-th classifier, we compute the margin gap between the largest and the mean of the others, i.e.

$$H = \max_{i=1}^{K} m_i - (\sum_{i=1}^{K} m_i - \max_{i=1}^{K} m_i)/(K-1) \qquad (1)$$

Where *K* denotes the total number of relation classes, and m_i denotes the output margin of the *i*-th classifier. Intuitively, the bigger the *H*, the greater the difference between the maximal margin and all others, and thus the more reliably the classifier makes the prediction on the instance.

```
**Algorithm** self-bootstrapping
Require: labeled seed set L
Require: unlabeled data set U
Require: batch size S
Repeat
    Train a single classifier on L
    Run the classifier on U
    Find at most S instances in U that the classifier has the highest prediction confidence
    Add them into L
Until: no data points available or the stoppage condition is reached
```

Figure 1: Self-bootstrapping algorithm

3.3 Clustering-based Stratified Seed Sampler

Stratified sampling is a method of sampling in statistics, in which the members of a population are grouped into relatively homogeneous subgroups (i.e. strata) according to one certain property, and then a sample is selected from each stratum. This process of grouping is called stratification, and the property on which the stratification is performed is called the stratification variable. Previous work justifies theoretically and practically that stratified sampling is more appropriate than random sampling for general use (Neyman, 1934) as well as for relation extraction (Qian et al., 2009). However, the difficulty lies in how to find the appropriate stratification variable for complicated tasks, such as relation extraction.

The idea of clustering-based stratification circumvents this problem by clustering the unlabeled data into a number of strata without the need to explicitly specify a stratification variable. Figure 2 illustrates the clustering-based stratified seed sampling strategy employed in the bootstrapping procedure, where *RSET* denotes the whole unlabeled data, *SeedSET* the seed set to be labeled and $|RSET_i|$ the number of instances in the *i*-th cluster[2] $RSET_i$. Here, a relation instance is represented using USPT and the similarity between two instances is computed using the standard convolution tree

[1] http://ai-nlp.info.uniroma2.it/moschitti/

[2] Hereafter, when we refer to clusters from the viewpoint of stratified sampling, they are often called "strata".

kernel, as described in Section 3.1 (i.e., both the clustering and the classification adopt the same structural representation, since we want the representative seeds in the clustering space to be also representative in the classification space). After clustering, a certain number of instances from every stratum are sampled using an intra-stratum scheme (c.f. Subsection 3.4). Normally, this number is proportional to the size of that stratum in the whole data set. However, in case this number is 0 due to the rounding of real numbers, it is set to 1 to ensure the existence of at least one seed from that stratum. Furthermore, to ensure that the total number of instances being sampled equals the prescribed N_S, the number of seeds from dominant strata may be slightly adjusted accordingly. Finally, these instances form the unlabeled seed set for an oracle to annotate as the input to the underlying supervised learner in the bootstrapping procedure.

3.4 Intra-stratum sampling

Given the distribution of clusters, a simple way to select the most representative instances is to choose the center of each cluster with the cluster prior as the weight of the center (Tang et al., 2002; Nguyen and Smeulders, 2004). Nevertheless, for the complicated task of relation extraction on the ACE RDC corpora, which is highly skewed across different relation classes, only considering the center of each cluster would severely under-represent the high-density data. To overcome this problem, we adopt a sampling approach, in particular stratified sampling, which takes the size of each stratum into consideration.

Given the size of the seed set N_S and the number of strata K, a natural question will arise as how to select the remaining (N_S-K) seeds after we have extracted the K centers from the K strata. We view this problem as intra-stratum sampling, which is required to choose the remaining number of seeds from inside individual stratum (excluding the centers themselves).

At the first glance, sampling a certain number of seeds from one particular stratum (e.g., $RSET_i$), seems to be the same sampling problem as we have encountered before, which aims to select the most representative and diverse seeds. This will naturally lead to another application of a clustering algorithm to the stratification of the stratum $RSET_i$.

```
Require: $RSET = \{R_1, R_2, \ldots, R_N\}$, the set of unlabeled relation instances and $K$, the number of strata being clustered
Output: $SeedSET$ with the size of $N_S$ (100)
Procedure
    Initialize $SeedSET = NULL$
    Cluster $RSET$ into $K$ strata using a clustering
        algorithm and perform stratum pruning if necessary.
    Calculate the number of instances being sampled
        for each stratum $i=\{1,2,\ldots,K\}$
            $N_i = \frac{|RSET_i|}{N} * N_S$        (2)
            and adjust this number if necessary.
    Perform intra-strata sampling to form $SeedSET_i$
        from each stratum $RSET_i$, by selecting the center
        $C_i$ and ($N_i$-1) additional instances
    Generate $SeedSET$ by summating $RSET_i$ from each
        stratum
```

Figure 2: Clustering-based stratified seed sampling

Nevertheless, remember the fact that, this time for the stratum $RSET_i$, the center C_i has been chosen, so it may not be reasonable to extract additional centers in this way. Therefore, in order to avoid recursion and over-complexity, we employ a diversity-motivated intra-stratum sampling scheme (Shen et al., 2004), called KDN (K-diverse neighbors), which aims to maximize the training utility of all seeds from a stratum. The motivation is that we prefer the seeds with high variance to each other, thus avoiding repetitious seeds from a single stratum. The basic idea is to add a candidate instance to the seed set only if it is sufficiently different from any previously selected seeds, i.e., the similarity between the candidate instance and any of the current seeds is less than a threshold β. In this paper, the threshold β is set to the average pair-wise similarity between any two instances in a stratum.

4 Clustering Algorithms

This section describes several typical clustering algorithms in the literature, such as K-means, HAC, spectral clustering and affinity propagation, as well as their application in this paper.

4.1 K-medoids (KM)

As a simple yet effective clustering method, the K-means algorithm assigns each instance to the cluster whose center (also called centroid) is nearest. In particular, the center is the average of all the instances in the cluster, i.e., with its coordinates the arithmetic means for each dimension separately over all the instances in the cluster.

One problem with K-means is that it does not yield the same result with each run while the other problem is the requirement for the concept of a mean to be definable, which is unfortunately not available in our setting (we employ a parse tree representation for a relation instance). Hence, we adopt a variant of K-means, namely, K-medoids, where a medoid, rather than a centroid, is defined as a representative of a cluster. Besides, K-medoids has proved to be more robust to noise and outliers in comparison with K-means.

4.2 Hierarchical Agglomerative Clustering (HAC)

Different from K-medoids, hierarchical clustering creates a hierarchy of clusters which can be represented in a tree structure called a dendrogram. The root of the tree consists of a single cluster containing all objects, and the leaves correspond to individual object.

Typically, hierarchical agglomerative clustering (HAC) starts at the leaves and successively merges two clusters together as long as they have the shortest distance among all the pair-wise distances between any two clusters.

Given a specified number of clusters, the key problem is to determine where to cut the hierarchical tree into clusters. In this paper, we generate the final flat cluster structures greedily by maximizing the equal distribution of instances among different clusters.

4.3 Spectral Clustering (SC)

Spectral clustering has become more and more popular recently. Taking as input a similarity matrix between any two instances, spectral clustering makes use of the spectrum of the similarity matrix of the data to perform dimensionality reduction for clustering in fewer dimensions.

Compared to the “traditional algorithms” such as K-means or HAC, spectral clustering has many fundamental advantages. Results obtained by spectral clustering often outperform the traditional approaches. Furthermore, spectral clustering is very simple to implement and can be solved efficiently using standard linear algebra methods (von Luxburg, 2006).

4.4 Affinity Propagation (AP)

As a new emerging clustering algorithm, affinity propagation (AP) (Frey and Dueck, 2007) is basically an iterative message-passing procedure in which the instances being clustered compete to serve as cluster exemplars by exchanging two types of messages, namely, “responsibility” and “availability”. After the procedure converges or has repeated a finite number of iterations, each cluster is represented by an exemplar. AP was reported to find clusters with much lower error than those found by other methods.

For our application, affinity propagation takes as input a similarity matrix, whose elements represent either the similarity between two different instances or the preference (a real number *p*) for an instance when two instances are the same. One problem with AP is that the number of clusters cannot be pre-defined, which is indirectly determined by the preference as well as the convergence procedure itself.

5 Experimentation

This section systematically evaluates the bootstrapping approach using clustering-based stratified seed sampling, in the relation classification (i.e., given the relationship already detected) subtask of relation extraction on the ACE RDC 2004 corpus.

5.1 Experimental Setting

The ACE RDC 2004 corpus[3] is gathered from various newspapers, newswire and broadcasts. It contains 451 documents and 5702 positive relation instances of 7 relation types and 23 subtypes between 7 entity types. For easy reference with related work in the literature, evaluation is done on 347 documents (from *nwire* and *bnews* domains), which include 4305 relation instances. Table 1 lists the major relation types and subtypes, including their corresponding instance numbers and ratios in our evaluation set. One obvious observation from the table is that the numbers of different relation types is highly imbalanced. These 347 documents are then divided into 3 disjoint sets randomly, with

[3] http//www.ldc.upenn.edu/ Projects/ACE/

Types	Subtypes	#	%
PHYS	Located	738	17.1
	Near	87	2.0
	Part-Whole	378	8.8
PER-SOC	Business	173	4.0
	Family	121	2.8
	Other	55	1.3
EMP-ORG	Employ-Executive	489	11.4
	Employ-Staff	539	12.5
	Employ-Undeter.	78	1.8
	Member-of-Group	191	4.4
	Subsidiary	206	4.8
	Partner	12	0.3
	Other	80	1.9
ART	User-or-Owner	200	4.6
	Inventor-or-Man.	9	0.2
	Other	2	0.0
OTHER-AFF	Ethnic	39	0.9
	Ideology	48	1.1
	Other	54	1.3
GPE-AFF	Citizen-or-Resid.	273	6.3
	Based-In	215	5.0
	Other	39	0.9
DISC		279	6.5
Total		4305	100.0

Table 1: Relation types and their corresponding instance numbers and ratios in the ACE RDC 2004 corpus

10% of them (35 files, around 400 instances) held out as the test data set, 10% of them (35 files, around 400 instances) used as the development data set to fine-tune various settings and parameters, while the remaining 277 files (over 3400 instances) used as the training data set, from which the seed set will be sampled.

The corpus is parsed using Charniak's parser (Charniak, 2001) and relation instances are generated by extracting all pairs of entity mentions occurring in the same sentence with positive relationships. For easy comparison with related work, we only evaluate the relation classification task on the 7 major relation types of the ACE RDC 2004 corpus. For the SVM^{light}-TK classifier, the training parameters C (SVM) and λ (tree kernel) are fine-tuned to 2.4 and 0.4 respectively.

The performance is measured using the standard P/R/F1 (Precision/Recall/F1-measure). For each relation type, P is the ratio of the true relation instances in all the relation instances being identified, R is the ratio of the true relation instances being identified in all the true relation instances in the corpus, and F1 is the harmonic mean of P and R. The overall performance P/R/F1 is then calculated using the micro-average measure over all major class types.

5.2 Experimental Results

Comparison of various seed sampling strategies without intra-stratum sampling on the development data

Table 2 compares the performance of bootstrapping-based relation classification using various seed sampling strategies without intra-stratum sampling on the development data. Here, the size of the seed set *L* is set to 100, and the top 100 instances with the highest confidence (c.f. Formula 1) are augmented at each iteration. For sampling strategies marked with an asterisk, we performed 10 trials and calculated their averages. Since for these strategies the seed sets sampled from different trials may be quite different, their performance scores vary in a great degree accordingly. This experimental setting and notation are also used in all the subsequent experiments unless specified. Besides, two additional baseline sampling strategies are included for comparison: sequential sampling (SEQ), which selects a sequentially-occurring *L* instances as the seed set, and random sampling (RAND), which randomly selects *L* instances as the seed set.

Table 2 shows that

1) RAND outperforms SEQ by 1.2 units in F1-score. This is due to the fact that the seed set via RAND may better reflect the distribution of the whole training data than that via SEQ, nevertheless at the expense of collecting the whole training data in advance.
2) While HAC performs moderately better than RAND, it is surprising that both KM and AP perform even worse than SEQ, and that SC performs worse than RAND. Furthermore, all the four clustering-based seed sampling strategies achieve much smaller performance improvement in F1-score than RAND, among which KM performs worst with performance improvement of only 0.1 in F1-score.

Sampling strategies	P(ΔP)	R(ΔR)	F1(ΔF1)
RAND*	69.1(3.1)	66.4(0.2)	67.8(2.0)
SEQ*	65.8(2.6)	68.0(0.1)	66.6(1.3)
KM*	62.0(0.9)	61.0(-0.5)	61.3(0.1)
HAC	**69.9(1.3)**	**70.4(0.4)**	**70.1(0.8)**
SC*	67.1(1.5)	68.1(0.0)	67.5(0.8)
AP	66.6(2.0)	66.2(0.1)	66.4(1.1)

Table 2: Comparison of various seed sampling strategies without intra-stratum sampling on the development data

3) All the performance improvements from bootstrapping largely come from the improvements in precision. While the bootstrapping procedure makes the SVM classifier more accurate, it lacks enough generalization ability.

To explain above special phenomena, we have a look at the clustering results. Our inspection reveals that most of them are severely imbalanced, i.e., some clusters are highly dense while others are extremely sparse. This indicates that merely selecting the centers from each cluster cannot properly represent the overall distribution. Moreover, the centers with high density lack the generalization ability due to its solitude in the cluster, leading to less performance enhancement than expected.

The only exception is HAC, which much outperforms RAND by 2.3 in F1-score, although HAC is usually not considered as an effective clustering algorithm. The reason may be that HAC creates a hierarchy of clusters in the top-down manner by cutting a cluster into two. Therefore, the centers in the two sibling clusters will be closer to each other than they are to the centers in other clusters. Besides, the final flat cluster structures given a special number of clusters are generated greedily from the cluster hierarchy by maximizing the equal distribution of instances among different clusters. In other words, when the cluster number reaches a certain threshold, the dense area will get more seeds represented in the seed set. As a consequence, the distribution of all the seeds sampled by HAC will approximate the distribution of the whole training data in some degree, while the seeds sampled by other clustering algorithm are kept as far as possible due to the objective of clustering and the lack of intra-stratum sampling.

These observations also justify the application of the stratified seed sampling to the bootstrapping procedure, which enforces the number of seeds sampled from a cluster to be proportional to its density, presumably approximated by its size in this paper.

Comparison of different cluster numbers with intra-stratum sampling on the development data

In order to fine-tune the optimal cluster numbers for seed sampling, we compare the performance of different numbers of clusters for each clustering algorithm on the development data set and report their F-scores in Table 3. For reference, we also list the F-score for golden clustering (GOLD), in which all instances are grouped in terms of their annotated ground relation major types (7), major types considering relation direction (13), subtypes (23), and subtypes considering direction (38). Besides, the performance of clustering-based semi-supervised relation classification is also measured over other typical cluster numbers (i.e., 1, 50, 60, 80, 100). Particularly, when the cluster number equals 1, it means that only diversity other than representativeness is considered in the seed sampling. Among these clustering algorithms, one of the distinct characteristics with the AP algorithm is that the number of clusters cannot be specified in advance, rather, it is determined by the pre-defined preference parameter (c.f. Subsection 4.4). Therefore, we should tune the preference parameter so as to get the pre-defined cluster number. However, sometimes we still couldn't get the exact number of clusters as we expected. In these cases, we use the approximate cluster numbers for AP instead.

Table 3 shows that

1) The performance for all the clustering algorithms varies in some degree with the number of clusters being grouped. Interestingly, the performance with only one cluster is better than those of clustering-based strategies with 100 clusters, at most cases. This implies that the diversity of the seeds is at least as important as their representativeness. And this could be further explained by our observation that, with the increase of cluster numbers, the clusters get smaller and denser while their centers also come closer to each other. Therefore, the representativeness and diversity as well as the distribution of the seeds sampled from them may vary accordingly, leading to different performance.

# of Clusters	GOLD	KM*	HAC	SC*	AP
1	-	68.7	68.7	-	-
7	**73.9**	70.3	**73.3**	**72.1**	-
13	70.2	68.9	70.3	67.3	-
23	64.9	**72.3**	72.9	68.9	71.1
38	60.8	69.9	71.6	68.0	**71.6**
50	-	68.5	69.9	68.5	70.4
60	-	66.3	68.5	68.6	69.7
80	-	64.2	65.9	68.0	68.1
100	-	61.3	70.1	67.5	66.4

Table 3: Performance in F1-score over different cluster numbers with intra-stratum sampling on the development data

2) Golden clustering achieves the best performance of 73.9 in F1-score when the cluster number is set to 7, significantly higher than the performance using other cluster numbers. Interestingly, this number coincides with the number of major relation types needed to be classified in our task. This is reasonable since the instances with the same relation type should be much more similar than those with different relation types and it is easy to discriminate the seed set of one relation type from that of other relation types.

3) Among the four clustering algorithms, HAC achieves best performance over most of cluster numbers. This further verifies the aforementioned analysis. That is, as a hierarchical clustering algorithm, HAC can sample seeds that better capture the distribution of the training data.

4) For KM, the best performance is achieved around the number of 23 while for both HAC and SC, the optimal cluster number is consistent with GOLD clustering, namely, 7. For AP, the optimal cluster number for AP is 38. This is largely due to that we fail to cluster the training data into about 7 and 13 groups no matter how we vary the preference parameter.

Final comparison of different clustering algorithms on the held-out test data

After the optimal cluster numbers are determined for each clustering algorithm, we apply these numbers on the held-out test data and report the performance results (P/R/F1 and their respective improvements) in Table 4. For easy reference, we also include the performance for GOLD, RAND, and SEQ sampling strategies.

Sampling strategies	P(ΔP)	R(ΔR)	F1(ΔF1)
GOLD	**79.5(7.8)**	72.7(2.1)	**76.0**(4.8)
RAND*	71.9(3.7)	69.7(0.1)	70.8(1.8)
SEQ*	71.9(2.6)	65.2(0.1)	69.3(1.3)
KM*	73.6(2.1)	72.3(0.3)	72.9(1.2)
HAC	79.0(10.2)	**73.0(1.1)**	**75.9(5.6)**
SC*	72.3(2.1)	72.1(0.4)	72.2(1.2)
AP	75.7(2.5)	72.0(0.4)	73.7(1.4)

Table 4: Performance of various clustering-based seed sampling strategies on the held-out test data with the optimal cluster number for each clustering algorithm

Table 4 shows that

1) Among all the clustering algorithms, HAC achieves the best F1-score of 75.9, significantly higher than RAND and SEQ by 5.1 and 6.6 respectively. The improvement comes not only from significant precision boost, but also from moderate recall increase. This further justifies the merits of HAC as a clustering algorithm for stratified seed sampling in semi-supervised relation classification.

2) HAC approaches the best F1-score of 76.0 for golden clustering. Obviously, this doesn't mean HAC performs as well as golden clustering in terms of clustering quality measures, rather it does imply that HAC achieves the performance improvement by making the seed set better represent the overall distribution over inherent structure of relation instances, while golden clustering accomplishes this using the distribution over relation types. Since the distribution over relation types doesn't always conform to that over instance structures, and for a statistical discriminative classifier, often the latter is more important than the former, it will be no surprise if HAC outperforms golden clustering in some real applications, e.g. clustering-based stratified sampling.

6 Conclusion and Future Work

This paper presents a stratified seed sampling strategy based on clustering algorithms for semi-supervised learning. Our strategy does not rely on any stratification variable to divide the training instances into a number of strata. Instead, the strata are formed via clustering, given a metric measuring the similarity between any two instances. Further, diversity-motivated intra-strata sampling is employed to sample additional instances from within each stratum besides its center. We compare the effect of various clustering algorithms on the performance of semi-supervised learning and find that HAC achieves the best performance since the distribution of its seed set better approximates that of the whole training data. Extensive evaluation on the ACE RDC 2004 benchmark corpus shows that our clustering-based stratified seed sampling strategy significantly improves the performance of semi-supervised relation classification.

We believe that our clustering-based stratified seed sampling strategy can not only be applied to other semi-supervised learning tasks, but also can be incorporated into active learning, where the instances to be labeled at each iteration as well as the seed set could be selected using clustering techniques, thus further reducing the amount of instances needed to be annotated.

For the future work, it is possible to adapt our one-level clustering-based sampling to the multi-level one, where for every stratum it is still possible to divide it into lower sub-strata for further stratified sampling in order to make the seeds better represent the true distribution of the data.

Acknowledgments

This research is supported by Projects 60873150, 60970056, and 90920004 under the National Natural Science Foundation of China.

References

S. Abney. 2002. Bootstrapping. *ACL-2002*.

E. Agichtein and L. Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the 5th ACM international Conference on Digital Libraries (ACMDL 2000)*.

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)*.

E. Charniak. 2001. Intermediate-head Parsing for Language Models. *ACL-2001*: 116-123.

M. Collins and N. Duffy. 2001. Convolution Kernels for Natural Language. *NIPS 2001*: 625-632.

J.X. Chen, D.H. Ji, C.L. Tan, and Z.Y. Niu. 2005. Unsupervised Feature Selection for Relation Extraction. *CIKM-2005*: 411-418.

J.X. Chen, D.H. Ji, and C. L. Tan. 2006. Relation Extraction using Label Propagation Based Semi supervised Learning. *ACL/COLING-2006*: 129-136.

A. Culotta and J. Sorensen. 2004. Dependency tree kernels for relation extraction. *ACL-2004*: 423-439.

B.J. Frey and D. Dueck. 2007. Clustering by Passing Messages between Data Points. *Science*, 315: 972-976.

T. Hasegawa, S. Sekine, and R. Grishman. 2004. Discovering Relations among Named Entities from Large Corpora. *ACL-2004*.

N. Kambhatla. 2004. Combining lexical, syntactic and semantic features with Maximum Entropy models for extracting relations. *ACL-2004(posters)*: 178-181.

S. Miller, H. Fox, L. Ramshaw, and R. Weischedel. 2000. A novel use of statistical parsing to extract information from text. In *Proceedings of the 6th Applied Natural Language Processing Conference*.

J. Neyman. 1934. On the Two Different Aspects of the Representative Method: The Method of Stratified Sampling and the Method of Purposive Selection. *Journal of the Royal Statistical Society*, 97(4): 558-625.

H.T. Nguyen and A. Smeulders. 2004. Active Learning Using Pre-clustering, *ICML*-2004.

L.H. Qian, G.D. Zhou, Q.M. Zhu, and P.D. Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. *COLING-2008*: 697-704.

L.H. Qian, G.D. Zhou, F. Kong, and Q.M. Zhu. 2009. Semi-Supervised Learning for Semantic Relation Classification using Stratified Sampling Strategy. *EMNLP-2009*: 1437-1445.

D. Shen, J. Zhang, J. Su, G. Zhou and C. Tan. 2004. Multi-criteria-based active learning for named entity recognition. *ACL*-2004.

M. Tang, X. Luo and S. Roukos. 2002. Active Learning for Statistical Natural Language Parsing. *ACL*-2002.

U. von Luxburg. 2006. A tutorial on spectral clustering. Technical report, Max Planck Institute for Biological Cybernetics.

D. Zelenko, C. Aone, and A. Richardella. 2003. Kernel Methods for Relation Extraction. *Journal of Machine Learning Research*, (2): 1083-1106.

M. Zhang, J. Su, D. M. Wang, G. D. Zhou, and C. L. Tan. 2005. Discovering Relations between Named Entities from a Large Raw Corpus Using Tree Similarity-Based Clustering. *IJCNLP-2005*: 378-389.

M. Zhang, J. Zhang, J. Su, and G.D. Zhou. 2006. A Composite Kernel to Extract Relations between Entities with both Flat and Structured Features. *ACL/COLING-2006*: 825-832.

Z. Zhang. 2004. Weakly-supervised relation classification for Information Extraction. *CIKM-2004*.

S.B. Zhao and R. Grishman. 2005. Extracting relations with integrated information using kernel methods. *ACL-2005*: 419-426.

G.D. Zhou, J. Su, J. Zhang, and M. Zhang. 2005. Exploring various knowledge in relation extraction. *ACL-2005*: 427-434.

G.D. Zhou, L.H. Qian, and J.X. Fan. 2010. Tree kernel-based semantic relation extraction with rich syntactic and semantic information. *Information Sciences*, (179): 1785-1791.

G.D. Zhou, L.H. Qian, and Q.M. Zhu. 2009. Label propagation via bootstrapped support vectors for semantic relation extraction between named entities. *Computer Speech and Language*, 23(4): 464-478.

G.D. Zhou and M. Zhang. 2007. Extraction relation information from text documents by exploring various types of knowledge. *Information Processing and Management*, (42):969-982.

G.D. Zhou, M. Zhang, D.H. Ji, and Q.M. Zhu. 2007. Tree Kernel-based Relation Extraction with Context-Sensitive Structured Parse Tree Information. *EMNLP/CoNLL-2007*: 728-736.

Unsupervised discovery of negative categories in lexicon bootstrapping

Tara McIntosh
NICTA Victoria Research Lab
Dept of Computer Science and Software Engineering
University of Melbourne
nlp@taramcintosh.org

Abstract

Multi-category bootstrapping algorithms were developed to reduce semantic drift. By extracting multiple semantic lexicons simultaneously, a category's search space may be restricted. The best results have been achieved through reliance on manually crafted negative categories. Unfortunately, identifying these categories is non-trivial, and their use shifts the unsupervised bootstrapping paradigm towards a supervised framework.

We present NEG-FINDER, the first approach for discovering negative categories automatically. NEG-FINDER exploits unsupervised term clustering to generate multiple negative categories during bootstrapping. Our algorithm effectively removes the necessity of manual intervention and formulation of negative categories, with performance closely approaching that obtained using negative categories defined by a domain expert.

1 Introduction

Automatically acquiring semantic lexicons from text is essential for overcoming the knowledge bottleneck in many NLP tasks, e.g. question answering (Ravichandran and Hovy, 2002). Many of the successful methods follow the unsupervised iterative bootstrapping framework (Riloff and Shepherd, 1997). Bootstrapping has since been effectively applied to extracting general semantic lexicons (Riloff and Jones, 1999), biomedical entities (Yu and Agichtein, 2003) and facts (Carlson et al., 2010).

Bootstrapping is often considered to be minimally supervised, as it is initialised with a small set of seed terms of the target category to extract. These seeds are used to identify patterns that can match the target category, which in turn can extract new lexicon terms (Riloff and Jones, 1999). Unfortunately, *semantic drift* often occurs when ambiguous or erroneous terms and/or patterns are introduced into the iterative process (Curran et al., 2007).

In multi-category bootstrapping, semantic drift is often reduced when the target categories compete with each other for terms and/or patterns (Yangarber et al., 2002). This process is most effective when the categories bound each other's search space. To ensure this, manually crafted negative categories are introduced (Lin et al., 2003; Curran et al., 2007). Unfortunately, this makes these algorithms substantially more supervised.

The design of negative categories is a very time consuming task. It typically requires a domain expert to identify the semantic drift and its cause, followed by a significant amount of trial and error in order to select the most suitable combination of negative categories. This introduces a substantial amount of supervised information into what was an unsupervised framework, and in turn negates one of the main advantages of bootstrapping — the quick construction of accurate semantic lexicons.

We show that although excellent performance is achieved using negative categories, it varies greatly depending on the negative categories selected. This highlights the difficulty of crafting negative categories and thus the necessity for tools that can automatically identify them.

Our second contribution is the first fully unsupervised approach, NEG-FINDER, for discovering

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 356–365,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

negative categories automatically. During bootstrapping, efficient clustering techniques are applied to sets of drifted candidate terms to generate new negative categories. Once a negative category is identified it is incorporated into the subsequent iterations whereby it provides the necessary semantic boundaries for the target categories.

We demonstrate the effectiveness of our approach for extracting biomedical semantic lexicons by incorporating NEG-FINDER within the WMEB-DRIFT bootstrapping algorithm (McIntosh and Curran, 2009). NEG-FINDER significantly outperforms bootstrapping prior to the domain expert's negative categories. We show that by using our discovered categories we can reach near expert-guided performance. Our methods effectively remove the necessity of manual intervention and formulation of negative categories in semantic lexicon bootstrapping.

2 Background

Various automated pattern-based bootstrapping algorithms have been proposed to iteratively build semantic lexicons. In multi-level bootstrapping, a lexicon is iteratively expanded from a small sample of seed terms (Riloff and Jones, 1999). The seed terms are used to identify contextual patterns they appear in, which in turn may be used to extract new lexicon entries. This process is repeated with the new expanded lexicon identifying new patterns.

When bootstrapping semantic lexicons, polysemous or erroneous terms and/or patterns that weakly constrain the semantic class are eventually extracted. This often causes *semantic drift* — when a lexicon's intended meaning shifts into another category during bootstrapping (Curran et al., 2007). For example, *female names* may drift into *gemstones* when the terms *Ruby* and *Pearl* are extracted.

Multi-category bootstrapping algorithms, such as BASILISK (Thelen and Riloff, 2002), NOMEN (Yangarber et al., 2002), and WMEB (McIntosh and Curran, 2008), aim to reduce semantic drift by extracting multiple semantic categories simultaneously. These algorithms utilise information about other semantic categories in order to reduce the categories from drifting towards each other. This framework has recently been extended to extract different relations from text (Carlson et al., 2010).

2.1 Weighted MEB

In Weighted Mutual Exclusion Bootstrapping (WMEB, McIntosh and Curran, 2008), multiple semantic categories iterate simultaneously between the term and pattern extraction phases, competing with each other for terms and patterns. Semantic drift is reduced by forcing the categories to be mutually exclusive. That is, candidate terms can only be extracted by a single category and patterns can only extract terms for a single category.

In WMEB, multiple bootstrapping instances are initiated for each competing target category. Each category's seed set forms its initial lexicon. For each term in the category lexicon, WMEB identifies all candidate contextual patterns that can match the term in the text. To ensure mutual exclusion between the categories, candidate patterns that are identified by multiple categories in an iteration are excluded. The remaining patterns are then ranked according to the *reliability measure* and *relevance weight*.

The reliability of a pattern for a given category is the number of extracted terms in the category's lexicon that match the pattern. A pattern's relevance weight is defined as the sum of the χ-squared values between the pattern (p) and each of the lexicon terms (t): $\mathrm{weight}(p) = \sum_{t \in T} \chi^2(p, t)$. These metrics are symmetrical for both candidate terms and patterns.

The top-m patterns are then added to the pool of extracting patterns. If each of the top-m patterns already exists in the pool, the next unseen pattern is added to the pool. This ensures at least one new pattern is added to the pool in each iteration.

In the term selection phase, all patterns within the pattern pool are used to identify candidate terms. Like the candidate patterns, terms that are extracted by multiple categories in the same iteration are also excluded. The remaining candidate terms are ranked with respect to their reliability and relevance weight, and the top-n terms are added to the lexicon.

2.2 Detecting semantic drift in WMEB

In McIntosh and Curran (2009), we showed that multi-category bootstrappers are still prone to semantic drift in the later iterations. We proposed a drift detection metric based on our hypothesis that semantic drift occurs when a candidate term is more similar to the recently added terms than to the seed and high precision terms extracted in the earlier iterations. Our metric is based on distributional similarity measurements and can be directly incorporated into WMEB's term selection phase to prevent drifting terms from being extracted (WMEB-DRIFT).

The drift metric is defined as the ratio of the average distributional similarity of the candidate term to the first n terms extracted into the lexicon L, and to the last m terms extracted in the previous iterations:

$$\mathrm{drift}(term, n, m) = \frac{\mathrm{avgsim}(L_{1...n}, term)}{\mathrm{avgsim}(L_{(N-m+1)...N}, term)} \quad (1)$$

2.3 Negative categories

In multi-category bootstrapping, improvements in precision arise when semantic boundaries between multiple target categories are established. Thus, it is beneficial to bootstrap categories that share similar semantic spaces, such as *female names* and *flowers*.

Unfortunately, it is difficult to predict if a target category will suffer from semantic drift and/or whether it will naturally compete with the other target categories. Once a domain expert establishes semantic drift and its possible cause, a set of *negative/stop* categories that may be of no direct interest are manually crafted to prevent semantic drift. These additional categories are then exploited during another round of bootstrapping to provide further competition for the target categories (Lin et al., 2003; Curran et al., 2007).

Lin et al. (2003) improved NOMEN's performance for extracting *diseases* and *locations* from the ProMED corpus by incorporating negative categories into the bootstrapping process. They first used one general negative category, seeded with the 10 most frequent nouns in the corpus that were unrelated to the target categories. This single negative category resulted in substantial improvements in precision. In their final experiment, six negative categories that were notable sources of semantic drift were identified, and the inclusion of these lead to further performance improvements (∼20%).

In similar experiments, both Curran et al. (2007) and McIntosh (2010) manually crafted negative categories that were necessary to prevent semantic drift. In particular, in McIntosh (2010), a biomedical expert spent considerable time (∼15 days) and effort identifying potential negative categories and subsequently optimising their associated seeds in trial and error bootstrapping runs.

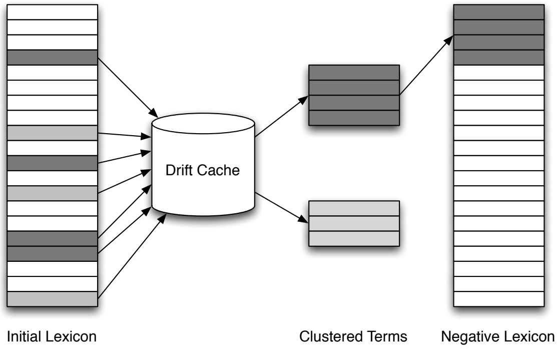


Figure 1: NEG-FINDER: Local negative discovery

By introducing manually crafted negative categories, a significant amount of expert domain knowledge is introduced. The use of this expert knowledge undermines the principle advantages of unsupervised bootstrapping, by making it difficult to bootstrap lexicons for a large number of categories across diverse domains or languages. In this paper, we aim to push multi-category bootstrapping back into its original minimally-supervised framework, with as little performance loss as possible.

3 NEG-FINDER

Our approach, *Negative Category Finder for Bootstrapping* (NEG-FINDER), can be easily incorporated into bootstrapping algorithms that exclude candidate terms or facts based on a selection criteria, including WMEB-DRIFT and Paşca et al.'s (2006) large-scale fact extraction system. For simplicity, we describe our approach within the WMEB-DRIFT bootstrapping algorithm. Figure 1 shows the framework of our approach.

To discover negative categories during bootstrapping, NEG-FINDER must identify a representative cluster of the drifted terms. In this section, we present the two types of clustering used (*maximum* and *outlier*), and our three different levels of negative discovery (*local*, *global* and *mixture*).

3.1 Discovering negative categories

We have observed that semantic drift begins to dominate when clusters of incorrect terms with similar meanings are extracted. In the term selection phase of WMEB-DRIFT, the top-n candidate terms that satisfy the drift detection threshold are added to the expanding lexicon. Those terms which are considered but do not meet the threshold are excluded.

In NEG-FINDER, these drifted terms are cached as they may provide adequate seed terms for new negative categories. However, the drifted terms can also include scattered polysemous or correct terms that share little similarity with the other drifted terms. Therefore, simply using the first set of drifted terms to establish a negative category is likely to introduce noise rather than a cohesive competing category.

To discover negative categories, we exploit hierarchical clustering to group similar terms within the cache of drifted terms. In agglomerative hierarchical clustering, a single term is assigned to an individual cluster, and these clusters are iteratively merged until a final cluster is formed containing all terms (Kaufmann and Rousseeuw, 1990). In our approach, the similarity between two clusters is computed as the average distributional similarity between all pairs of terms across the clusters (average-link clustering).

For calculating the similarity between two terms we use the distributional similarity approach described in Curran (2004). We extracted window-based features from the set of candidate patterns to form context vectors for each term. We use the standard t-test weight and weighted Jaccard measure functions (Curran, 2004).

To ensure adequate coverage of the possible drifting topics, negative discovery and hence clustering is only performed when the drift cache consists of at least 20 terms.

3.2 Maximum and outlier clustering

Although hierarchical clustering is quadratic, we can efficiently exploit the agglomerative process as the most similar terms will merge into clusters first. Therefore, to identify the k most similar terms, we can exit the clustering process as soon as a cluster of size k is established. We refer to this approach as *maximum clustering*.

In our next clustering method, we aim to form a negative category with as little similarity to the target seeds. We use an *outlier clustering* strategy, in which the drifted term t with the least average distributional similarity to the first n terms in the lexicon must be contained in the cluster of seeds. We use average similarity to the first n terms, as it is already pre-computed for the drift detection metric. As with *maximum clustering*, once a cluster of size k containing the term t is formed, the clustering process can be terminated.

3.3 Incorporating the negative category

After a cluster of negative seed terms is established, the drift cache is cleared, and a new negative category is created and introduced into the iterative bootstrapping process in the next iteration. This means that the negative category can only influence the subsequent iterations of bootstrapping. The negative categories can compete with all other categories, including any previously introduced negative categories, however the negative categories do not contribute to the drift caches.

Before the new category is introduced, its first set of extracting patterns must be identified. For this, the complete set of extracting patterns matching any of the negative seeds are considered and ranked with respect to the seeds. The top scoring patterns are considered sequentially until m patterns are assigned to the new negative category. To ensure mutual exclusion between the new category and the target categories, a candidate pattern that has previously been selected by a target category cannot be used to extract terms for either category in the subsequent iterations.

3.4 Levels of negative discovery

Negative category discovery can be performed at a *local* or *global* level, or as a *mixture* of both. In *local discovery*, each target category has its own drifted term cache and can generate negative categories irrespective of the other target categories. This is shown in Figure 1. The drifted terms (shaded) are extracted away from the lexicon into the local drift cache, which is then clustered. A cluster is then used to initiate a negative category's lexicon. Target categories can also generate multiple negative categories across different iterations.

In *global discovery*, all drifted terms are pooled into a global cache, from which a single negative category can be identified in an iteration. This is based on our intuition that multiple target categories

TYPE	MEDLINE
No. Terms	1 347 002
No. Patterns	4 090 412
No. 5-grams	72 796 760
No. Unfiltered tokens	6 642 802 776

Table 1: Filtered 5-gram dataset statistics.

may be drifting into similar semantic categories, and enables these otherwise missed negative categories to be established.

In the *mixture discovery* method, both global and local negative categories can be formed. A category's drifted terms are collected into its local cache as well as the global cache. Negative discovery is then performed on each cache when they contain at least 20 terms. Once a local negative category is formed, the terms within the local cache are cleared and also removed from the global cache. This prevents multiple negative categories being instantiated with overlapping seed terms.

4 Experimental setup

To compare the effectiveness of our negative discovery approaches we consider the task of extracting biomedical semantic lexicons from raw text.

4.1 Data

The algorithms take as input a set of candidate terms to be extracted into semantic lexicons. The source text collection consists of 5-grams (t_1, t_2, t_3, t_4, t_5) from approximately 16 million MEDLINE abstracts.[1] The set of possible candidate terms correspond to the middle tokens (t_3), and the possible patterns are formed from the surrounding tokens (t_1, t_2, t_4, t_5). We do not use syntactic knowledge, as we did not wish to rely on any tools that require supervised training, to ensure our technique is as domain and language independent as possible.

Limited preprocessing was required to extract the 5-grams from MEDLINE. The XML markup was removed, and the collection was tokenised and split into sentences using bio-specific NLP tools (Grover et al., 2006). Filtering was applied to remove infrequent patterns and terms – patterns appearing with less than 7 different terms, and terms only appearing with those patterns were removed. The statistics of the resulting dataset are shown in Table 1.

CAT	DESCRIPTION
ANTI	Antibodies: *MAb IgG IgM rituximab infliximab* (κ_1:0.89, κ_2:1.0)
CELL	Cells: *RBC HUVEC BAEC VSMC SMC* (κ_1:0.91, κ_2:1.0)
CLNE	Cell lines: *PC12 CHO HeLa Jurkat COS* (κ_1:0.93, κ_2: 1.0)
DISE	Diseases: *asthma hepatitis tuberculosis HIV malaria* (κ_1:0.98, κ_2:1.0)
DRUG	Drugs: *acetylcholine carbachol heparin penicillin tetracyclin* (κ_1:0.86, κ_2:0.99)
FUNC	Molecular functions and processes: *kinase ligase acetyltransferase helicase binding* (κ_1:0.87, κ_2:0.99)
MUTN	Protein and gene mutations: *Leiden C677T C282Y 35delG null* (κ_1:0.89, κ_2:1.0)
PROT	Proteins and genes: *p53 actin collagen albumin IL-6* (κ_1:0.99, κ_2:1.0)
SIGN	Signs and symptoms: *anemia fever hypertension hyperglycemia cough* (κ_1:0.96, κ_2:0.99)
TUMR	Tumors: *lymphoma sarcoma melanoma osteosarcoma neuroblastoma* (κ_1:0.89, κ_2:0.95)

Table 2: The MEDLINE semantic categories

4.2 Semantic categories

The semantic categories we extract from MEDLINE were inspired by the TREC Genomics entities (Hersh et al., 2007) and are described in detail in McIntosh (2010). The hand-picked seeds selected by a domain expert for each category are shown in italics in Table 2. These were carefully chosen to be as unambiguous as possible with respect to the other categories.

4.3 Negative categories

In our experiments, we use two different sets of negative categories. These are shown in Table 3. The first set corresponds to those used in McIntosh and Curran (2008), and were identified by a domain expert as common sources of semantic drift in preliminary experiments with MEB and WMEB. The AMINO ACID category was created in order to filter common MUTN errors. The ANIMAL and BODY PART categories were formed with the intention of preventing drift in the CELL, DISE and SIGN categories. The ORGANISM category was then created to reduce the new drift forming in the DISE category after the first set of negative categories were introduced.

The second set of negative categories was identified by an independent domain expert with limited

[1] The set contains all MEDLINE titles and abstracts available up to Oct 2007.

CATEGORY	SEED TERMS
1 AMINO ACID	arginine cysteine glycine glutamate histamine
ANIMAL	insect mammal mice mouse rats
BODY PART	breast eye liver muscle spleen
ORGANISM	Bartonella Borrelia Cryptosporidium Salmonella toxoplasma
2 AMINO ACID	Asn Gly His Leu Valine
ANIMAL	animals dogs larvae rabbits rodents
ORGANISM	Canidia Shigella Scedosporium Salmonella Yersinia
GENERIC	decrease effects events increase response
MODIFIERS	acute deep intrauterine postoperative secondary
PEOPLE	children females men subjects women
SAMPLE	biopsies CFU sample specimens tissues

Table 3: Manually crafted negative categories

knowledge of NLP and bootstrapping. This expert identified three similar categories to the first expert, however their seeds are very different. They also identified three more categories than the first.

4.4 Lexicon evaluation

Our evaluation process follows that of McIntosh and Curran (2009) and involved manually inspecting each extracted term and judging whether it was a member of the semantic class. This manual evaluation was performed by two domain experts and is necessary due to the limited coverage of biomedical resources. Inter-annotator agreement scores are provided in Table 2.[2] To make later evaluations more efficient, all evaluators' decisions for each category are cached.

Unfamiliar terms were checked using online resources including MEDLINE, MeSH, and Wikipedia. Each ambiguous term was counted as correct if it was classified into one of its correct categories, such as *lymphoma*, which is a TUMR and DISE. If a term was unambiguously part of a multi-word term we considered it correct. Abbreviations, acronyms, and obvious misspelled words were included.

For comparing the performance of the algorithms, the average precision for the top-1000 terms over the 10 target categories is measured. To identify when semantic drift has a significant impact, we report the precision of specific sections of the lexicon, e.g. the 801-1000 sample corresponds to the last 200 terms.

	1-500	1-1000
WMEB-DRIFT	74.3	68.6
+negative 1	87.7	82.8
+negative 2	83.8	77.8

Table 4: Influence of negative categories

4.5 System settings

All experiments were performed using the 10 target categories as input. Unless otherwise stated, no hand-picked negative categories are used.

Each target category is initialised with the 5 hand-picked seed terms (Table 2). In each iteration a maximum of 5 lexicon terms and 5 new patterns can be extracted by a category. The bootstrapping algorithms are run for 200 iterations.

The drift detection metric is calculated over the first 100 terms and previous 5 terms extracted into the lexicon, and the filter threshold is set to 0.2, as in McIntosh and Curran (2009). To ensure infrequent terms are not used to seed negative categories, drifted terms must occur at least 50 times to be retained in the drift cache. Negative category discovery is only initiated when the drifted cache contains at least 20 terms, and a minimum of 5 terms are used to seed a negative category.

4.6 Random seed experiments

Both McIntosh and Curran (2009) and Pantel et al. (2009) have shown that a bootstrapper's performance can vary greatly depending on the input seeds. To ensure our methods are compared reliably, we also report the average precision of randomised seed experiments. Each algorithm is instantiated 10 times with different random gold seeds for each target category. These gold seeds are randomly sampled from the evaluation cache formed in McIntosh and Curran (2009).

5 Results

5.1 Influence of negative categories

In our first experiments, we investigate the performance variations and improvements gained using negative categories selected by two independent domain experts. Table 4 shows WMEB-DRIFT's average precision over the 10 target categories with and without the two negative category sets. Both

[2] All disagreements were discussed, and the kappa scores κ_1 and κ_2 are those before and after the discussions, respectively.

	1-200	201-400	401-600	601-800	801-1000	1-1000
WMEB-DRIFT	79.5	74.8	64.7	61.9	62.1	68.6
NEG-FINDER						
First discovered	79.5	74.3	64.8	67.8	66.6	70.7
Local discovery						
+maximum	79.5	74.8	67.3	69.3	70.5	72.2
+outlier	79.5	73.9	64.8	67.8	71.0	71.5
Global discovery						
+maximum	79.5	73.9	65.7	73.2	72.7	73.4
+outlier	79.5	74.7	65.6	71.4	68.2	72.1
Mixture discovery						
+maximum	79.5	74.7	69.3	73.3	72.8	74.0
+outlier	79.5	75.2	69.7	72.0	69.4	73.2

Table 5: Performance comparison of WMEB-DRIFT and NEG-FINDER

sets significantly improve WMEB-DRIFT, however there is a significant performance difference between them. This demonstrates the difficulty of selecting appropriate negative categories and seeds for the task, and in turn the necessity for tools to discover them automatically.

5.2 Negative category discovery

Table 5 compares the performance of NEG-FINDER incorporated with WMEB-DRIFT. Each method has equal average precision over the first 200 terms, as semantic drift does not typically occur in the early iterations. Each discovery method significantly outperforms WMEB-DRIFT in the later stages, and over the top 1000 terms.[3]

The *first discovery* approach corresponds to the naïve NEG-FINDER system that generates *local* negative categories from the first five drifted terms. Although it outperforms WMEB-DRIFT, its advantage is smaller than the clustering methods.

The *outlier clustering* approach, which we predicted to be the most effective, was surprisingly less accurate than the *maximum* approach for selecting negative seeds. This is because the seed cluster formed around the outlier term is not guaranteed to have high pair-wise similarity and thus it may represent multiple semantic categories.

Local discovery was the least effective discovery approach. Compared to local discovery, *global discovery* is capable of detecting new negative categories earlier, and the categories it detects are more likely to compete with multiple target categories.

CATEGORY	NEGATIVE SEEDS
CELL-NEG	animals *After* Lambs Pigs Rabbits
TUMR-NEG	inoperable multinodular nonresectable operated unruptured
GLOBAL	days Hz mM post Torr
GLOBAL	aortas eyes legs mucosa retinas
GLOBAL	men offspring parents persons relatives
GLOBAL	Australian Belgian Dutch European Italian
GLOBAL	Amblyospora Branhamella Phormodium Pseudanabaena Rhodotorula

Table 6: Negative categories from mixture discovery

The NEG-FINDER *mixture* approach, which benefits from both *local* and *global discovery*, identifies the most useful negative categories. Table 6 shows the seven discovered categories — two local negative categories from CELL and TUMOUR, and five global categories were formed. Many of these categories are similar to those identified by the domain experts. For example, clear categories for ANIMAL, BODY PART, PEOPLE and ORGANISM are created. By identifying and then including these negative categories, NEG-FINDER significantly outperforms WMEB-DRIFT by 5.4% over the top-1000 terms and by 10.7% over the last 200 terms, where semantic drift is prominent. These results demonstrate that suitable negative categories can be identified and exploited during bootstrapping.

5.3 Boosting hand-picked negative categories

In our next set of experiments, we investigate whether NEG-FINDER can improve state-of-the-art performance by identifying new negative categories in addition to the manually selected negative

[3] Statistical significance was tested using computationally-intensive randomisation tests (Cohen, 1995).

	1-200	201-400	401-600	601-800	801-1000	1-1000
WMEB-DRIFT						
+negative 1	90.5	87.3	82.0	74.6	79.8	82.8
+negative 2	87.8	82.2	78.7	76.1	63.3	77.8
WMEB-DRIFT						
+restart +local	85.5	82.6	76.5	75.7	68.5	78.4
+restart +global	84.0	83.8	79.1	74.8	69.5	79.7
+restart +mixture	85.2	85.0	82.3	72.5	72.7	81.4

Table 7: Performance of WMEB-DRIFT using negative categories discovered by NEG-FINDER

	601-800	801-1000	1-1000
WMEB-DRIFT			
+negative 1	74.6	79.8	82.8
NEG-FINDER			
+negative 1 +local	76.4	80.1	83.2
+negative 1 +global	77.5	76.0	82.7
+negative 1 +mixture	76.7	79.9	83.2

Table 8: Performance of NEG-FINDER with manually crafted negative categories

categories. Both NEG-FINDER and WMEB-DRIFT are initialised with the 10 target categories and the first set of negative categories.

Table 8 compares our best performing systems (NEG-FINDER *maximum clustering*) with standard WMEB-DRIFT, over the last 400 terms where semantic drift dominates. NEG-FINDER effectively discovers additional categories and significantly outperforms WMEB-DRIFT. This further demonstrates the utility of our approach.

5.4 Restarting with new negative categories

The performance improvements so far using NEG-FINDER have been limited by the time at which new negative categories are discovered and incorporated into the bootstrapping process. That is, system improvements can only be gained from the negative categories after they are generated. For example, in *Local* NEG-FINDER, five negative categories are discovered in iterations 83, 85, 126, 130 and 150. On the other hand, in the WMEB-DRIFT +negative experiments (Table 8 row 2), the hand-picked negative categories can start competing with the target categories in the very first iteration of bootstrapping.

To test the full utility of NEG-FINDER, we use the set of discovered categories as competing input for WMEB-DRIFT. Table 7 shows the average precision of WMEB-DRIFT over the 10 target categories when it is restarted with the new negative categories discovered from our three approaches (using *maximum clustering*). Over the first 200 terms, significant improvements are gained using the new negative categories (+6%). However, the manually selected categories are far superior in preventing drift (+11%). This may be attributed by the target categories not strongly drifting into the new negative categories until the later stages, whereas the hand-picked categories were selected on the basis of observed drift in the early stages (over the first 500 terms).

Each NEG-FINDER approach significantly outperforms WMEB-DRIFT with no negative categories. For example, using the NEG-FINDER *mixture* categories increases precision by 12.8%. These approaches also outperform their corresponding inline discovery methods (e.g. +7.4% with *mixture discovery* – Table 5).

Table 7 shows that each of the discovered negative sets can significantly outperform the negative categories selected by a domain expert (negative set 2) (+0.6 – 3.9%). Our best system's performance (*mixture*: 81.4%) closely approaches that of the superior negative set, trailing by only 1.4%.

5.5 Individual categories

In this section, we analyse the effect of NEG-FINDER on the individual target categories. Table 9 shows the average precision of the lexicons for some target categories. All categories, except TUMOUR, improve significantly with the inclusion of the discovered negative categories. In particular, the CELL and SIGN categories, which are affected severely by semantic drift, increase by up to 33.3% and 45.2%, respectively. The discovered negative categories are more effective than the manually crafted sets in reducing semantic drift in the ANTIBODY, CELL and DISEASE lexicons.

	ANTI	CELL	DISE	SIGN	TUMR
WMEB-DRIFT	92.9	47.8	49.3	27.9	39.5
+negative 1	91.6	73.1	87.8	76.5	48.7
+negative 2	85.8	68.0	84.2	71.3	16.3
NEG-FINDER					
+mixture	94.9	73.9	56.0	41.0	42.2
+mixture +negative 1	90.8	77.2	87.8	78.2	48.2
WMEB-DRIFT					
+restart +local	89.9	78.8	71.6	73.1	32.2
+restart +global	94.6	79.0	81.9	62.6	35.2
+restart +mixture	92.6	81.1	91.1	63.6	47.5

Table 9: Individual category results (1-1000 terms)

5.6 Random seed experiments

In Table 10, we report the results of our randomised experiments. Over the last 200 terms, WMEB-DRIFT with the first set of negative categories (row 2) is outperformed by NEG-FINDER (row 4). NEG-FINDER also significantly boosts the performance of the original negative categories by identifying additional negative categories (row 5). Our final experiment, where WMEB-DRIFT is re-initialised with the negative categories discovered by NEG-FINDER, further demonstrates the utility of our method. On average, the discovered negative categories significantly outperform the manually crafted negative categories.

6 Conclusion

In this paper, we have proposed the first completely unsupervised approach to identifying the negative categories that are necessary for bootstrapping large yet precise semantic lexicons. Prior to this work, negative categories were manually crafted by a domain expert, undermining the advantages of an unsupervised bootstrapping paradigm.

There are numerous avenues for further examination. We intend to use sophisticated clustering methods, such as CBC (Pantel, 2003), to identify multiple negative categories across the target categories in a single iteration. We would also like to explore the suitability of NEG-FINDER for relation extraction.

Our initial analysis demonstrated that although excellent performance is achieved using negative categories, large performance variations occur when using categories crafted by different domain experts.

In NEG-FINDER, unsupervised clustering approaches are exploited to automatically discover negative categories during bootstrapping. NEG-FINDER identifies cohesive negative categories and many of these are semantically similar to those identified by domain experts.

	401-600	801-1000
WMEB-DRIFT	66.9	58.5
+negative 1	73.1	61.7
NEG-FINDER		
+mixture	71.9	64.2
+mixture +negative 1	76.1	66.7
WMEB-DRIFT		
+restart +mixture	78.0	70.8

Table 10: Random seed results

NEG-FINDER significantly outperforms the state-of-the-art algorithm WMEB-DRIFT, before negative categories are crafted, by up to 5.4% over the top-1000 terms; and by 10.7% over the last 200 terms extracted, where semantic drift is extensive. The new discovered categories can also be fully exploited in bootstrapping, where they successfully outperform a domain expert's negative categories and approach that of another expert.

The result is an effective approach that can be incorporated within any bootstrapper. NEG-FINDER successfully removes the necessity of including manually crafted supervised knowledge to boost a bootstrapper's performance. In doing so, we revert the multi-category bootstrapping framework back to its originally intended minimally supervised framework, with little performance trade-off.

Acknowledgements

We would like to thank Dr Cassie Thornley, our second evaluator; and the anonymous reviewers for their helpful feedback. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

Andrew Carlson, Justin Betteridge, Richard C. Wang, Jr. Estevam R. Hruschka, and Tom M. Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 101–110, New York, NY, USA.

Paul R. Cohen. 1995. *Empirical Methods for Artificial Intelligence*. MIT Press, Cambridge, MA, USA.

James R. Curran, Tara Murphy, and Bernhard Scholz. 2007. Minimising semantic drift with mutual exclusion bootstrapping. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, pages 172–180, Melbourne, Australia.

James R. Curran. 2004. *From Distributional to Semantic Similarity*. Ph.D. thesis, University of Edinburgh, Edinburgh, UK.

Claire Grover, Michael Matthews, and Richard Tobin. 2006. Tools to address the interdependence between tokenisation and standoff annotation. In *Proceedings of the 5th Workshop on NLP and XML: Multi-Dimensional Markup in Natural Language Processing*, pages 19–26, Trento, Italy.

William Hersh, Aaron M. Cohen, Lynn Ruslen, and Phoebe M. Roberts. 2007. TREC 2007 Genomics track overview. In *Proceedings of the 16th Text REtrieval Conference*, Gaithersburg, MD, USA.

Leonard Kaufmann and Peter J. Rousseeuw. 1990. *Finding Groups in Data: an Introdution to Cluster Analysis*. John Wiley and Sons.

Winston Lin, Roman Yangarber, and Ralph Grishman. 2003. Bootstrapped learning of semantic classes from positive and negative examples. In *Proceedings of the ICML-2003 Workshop on The Continuum from Labeled to Unlabeled Data*, pages 103–111, Washington, DC, USA.

Tara McIntosh and James R. Curran. 2008. Weighted mutual exclusion bootstrapping for domain independent lexicon and template acquisition. In *Proceedings of the Australasian Language Technology Association Workshop*, pages 97–105, Hobart, Australia.

Tara McIntosh and James R. Curran. 2009. Reducing semantic drift with bagging and distributional similarity. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 396–404, Suntec, Singapore.

Tara McIntosh. 2010. *Reducing Semantic Drift in Biomedical Lexicon Bootstrapping*. Ph.D. thesis, University of Sydney.

Marius Paşca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. 2006. Names and similarities on the web: Fact extraction in the fast lane. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 809–816, Sydney, Australia.

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 238–247, Singapore, Singapore.

Patrick Pantel. 2003. *Clustering by Committee*. Ph.D. thesis, University of Alberta.

Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 41–47, Philadelphia, PA, USA.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the 16th National Conference on Artificial Intelligence and the 11th Innovative Applications of Artificial Intelligence Conference*, pages 474–479, Orlando, FL, USA.

Ellen Riloff and Jessica Shepherd. 1997. A corpus-based approach for building semantic lexicons. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 117–124, Providence, RI, USA.

Michael Thelen and Ellen Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 214–221, Philadelphia, PA, USA.

Roman Yangarber, Winston Lin, and Ralph Grishman. 2002. Unsupervised learning of generalized names. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING)*, pages 1135–1141, San Francisco, CA, USA.

Hong Yu and Eugene Agichtein. 2003. Extracting synonymous gene and protein terms from biological literature. *Bioinformatics*, 19(1):i340–i349.

Automatic Keyphrase Extraction via Topic Decomposition

Zhiyuan Liu, Wenyi Huang, Yabin Zheng and **Maosong Sun**
Department of Computer Science and Technology
State Key Lab on Intelligent Technology and Systems
National Lab for Information Science and Technology
Tsinghua University, Beijing 100084, China
{lzy.thu, harrywy, yabin.zheng}@gmail.com,
sms@tsinghua.edu.cn

Abstract

Existing graph-based ranking methods for keyphrase extraction compute a *single* importance score for each word via a *single* random walk. Motivated by the fact that both documents and words can be represented by a mixture of semantic topics, we propose to decompose traditional random walk into multiple random walks specific to various topics. We thus build a Topical PageRank (TPR) on word graph to measure word importance with respect to different topics. After that, given the topic distribution of the document, we further calculate the ranking scores of words and extract the top ranked ones as keyphrases. Experimental results show that TPR outperforms state-of-the-art keyphrase extraction methods on two datasets under various evaluation metrics.

1 Introduction

Keyphrases are defined as a set of terms in a document that give a brief summary of its content for readers. Automatic keyphrase extraction is widely used in information retrieval and digital library (Turney, 2000; Nguyen and Kan, 2007). Keyphrase extraction is also an essential step in various tasks of natural language processing such as document categorization, clustering and summarization (Manning and Schutze, 2000).

There are two principled approaches to extracting keyphrases: supervised and unsupervised. The supervised approach (Turney, 1999) regards keyphrase extraction as a classification task, in which a model is trained to determine whether a candidate phrase is a keyphrase. Supervised methods require a document set with human-assigned keyphrases as training set. In Web era, articles increase exponentially and change dynamically, which demands keyphrase extraction to be efficient and adaptable. However, since human labeling is time consuming, it is impractical to label training set from time to time. We thus focus on the unsupervised approach in this study.

In the unsupervised approach, graph-based ranking methods are state-of-the-art (Mihalcea and Tarau, 2004). These methods first build a word graph according to word co-occurrences within the document, and then use random walk techniques (e.g., PageRank) to measure word importance. After that, top ranked words are selected as keyphrases.

Existing graph-based methods maintain a *single* importance score for each word. However, a document (e.g., news article or research article) is usually composed of multiple semantic topics. Taking this paper for example, it refers to two major topics, “keyphrase extraction” and “random walk”. As words are used to express various meanings corresponding to different semantic topics, a word will play different importance roles in different topics of the document. For example, the words “phrase” and “extraction” will be ranked to be more important in topic “keyphrase extraction”, while the words “graph” and “PageRank” will be more important in topic “random walk”. Since they do not take topics into account, graph-based methods may suffer from the following two problems:

1. Good keyphrases should be relevant to the major topics of the given document. In graph-based methods, the words that are strongly connected with other words tend to be ranked high,

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 366–376,
MIT, Massachusetts, USA, 9-11 October 2010.

which do not necessarily guarantee they are relevant to major topics of the document.

2. An appropriate set of keyphrases should also have a good coverage of the document's major topics. In graph-based methods, the extracted keyphrases may fall into a single topic of the document and fail to cover other substantial topics of the document.

To address the problem, it is intuitive to consider the topics of words and document in random walk for keyphrase extraction. In this paper, we propose to decompose traditional PageRank into multiple PageRanks specific to various topics and obtain the importance scores of words under different topics. After that, with the help of the document topics, we can further extract keyphrases that are relevant to the document and at the same time have a good coverage of the document's major topics. We call the topic-decomposed PageRank as Topical PageRank (TPR).

In experiments we find that TPR can extract keyphrases with high relevance and good coverage, which outperforms other baseline methods under various evaluation metrics on two datasets. We also investigate the performance of TPR with different parameter values and demonstrate its robustness. Moreover, TPR is unsupervised and language-independent, which is applicable in Web era with enormous information.

TPR for keyphrase extraction is a two-stage process:

1. Build a topic interpreter to acquire the topics of words and documents.

2. Perform TPR to extract keyphrases for documents.

We will introduce the two stages in Section 2 and Section 3.

2 Building Topic Interpreters

To run TPR on a word graph, we have to acquire topic distributions of words. There are roughly two approaches that can provide topics of words: (1) Use manually annotated knowledge bases, e.g., WordNet (Miller et al., 1990); (2) Use unsupervised machine learning techniques to obtain word topics from a large-scale document collection. Since the vocabulary in WordNet cannot cover many words in modern news and research articles, we employ the second approach to build topic interpreters for TPR.

In machine learning, various methods have been proposed to infer latent topics of words and documents. These methods, known as latent topic models, derive latent topics from a large-scale document collection according to word occurrence information. Latent Dirichlet Allocation (LDA) (Blei et al., 2003) is a representative of topic models. Compared to Latent Semantic Analysis (LSA) (Landauer et al., 1998) and probabilistic LSA (pLSA) (Hofmann, 1999), LDA has more feasibility for inference and can reduce the risk of over-fitting.

In LDA, each word w of a document d is regarded to be generated by first sampling a topic z from d's topic distribution $\theta^{(d)}$, and then sampling a word from the distribution over words $\phi^{(z)}$ that characterizes topic z. In LDA, $\theta^{(d)}$ and $\phi^{(z)}$ are drawn from conjugate Dirichlet priors α and β, separately. Therefore, θ and ϕ are integrated out and the probability of word w given document d and priors is represented as follows:

$$pr(w|d,\alpha,\beta) = \sum_{z=1}^{K} pr(w|z,\beta) pr(z|d,\alpha), \quad (1)$$

where K is the number of topics.

Using LDA, we can obtain the topic distribution of each word w, namely $pr(z|w)$ for topic $z \in K$. The word topic distributions will be used in TPR. Moreover, using the obtained word topic distributions, we can infer the topic distribution of a new document (Blei et al., 2003), namely $pr(z|d)$ for each topic $z \in K$, which will be used for ranking keyphrases.

3 Topical PageRank for Keyphrase Extraction

After building a topic interpreter to acquire the topics of words and documents, we can perform keyphrase extraction for documents via TPR. Given a document d, the process of keyphrase extraction using TPR consists of the following four steps which is also illustrated in Fig. 1:

1. Construct a word graph for d according to word co-occurrences within d.

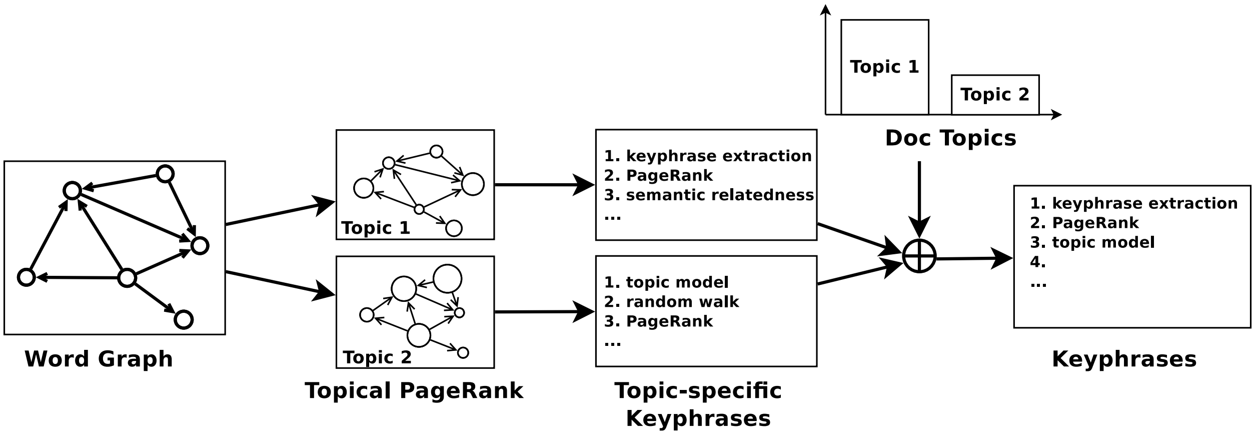


Figure 1: Topical PageRank for Keyphrase Extraction.

2. Perform TPR to calculate the importance scores for each word with respect to different topics.

3. Using the topic-specific importance scores of words, rank candidate keyphrases respect to each topic separately.

4. Given the topics of document d, integrate the topic-specific rankings of candidate keyphrases into a final ranking, and the top ranked ones are selected as keyphrases.

3.1 Constructing Word Graph

We construct a word graph according to word co-occurrences within the given document, which expresses the cohesion relationship between words in the context of document. The document is regarded as a word sequence, and the link weights between words is simply set to the co-occurrence count within a sliding window with maximum W words in the word sequence.

It was reported in (Mihalcea and Tarau, 2004) the graph direction does not influence the performance of keyphrase extraction very much. In this paper we simply construct word graphs with directions. The link directions are determined as follows. When sliding a W-width window, at each position, we add links from the first word pointing to other words within the window. Since keyphrases are usually noun phrases, we only add adjectives and nouns in word graph.

3.2 Topical PageRank

Before introducing TPR, we first give some formal notations. We denote $G = (V, E)$ as the graph of a document, with vertex set $V = \{w_1, w_2, \cdots, w_N\}$ and link set $(w_i, w_j) \in E$ if there is a link from w_i to w_j. In a word graph, each vertex represents a word, and each link indicates the relatedness between words. We denote the weight of link (w_i, w_j) as $e(w_i, w_j)$, and the out-degree of vertex w_i as $O(w_i) = \sum_{j:w_i \rightarrow w_j} e(w_i, w_j)$.

Topical PageRank is based on PageRank (Page et al., 1998). PageRank is a well known ranking algorithm that uses link information to assign global importance scores to web pages. The basic idea of PageRank is that a vertex is important if there are other important vertices pointing to it. This can be regarded as voting or recommendation among vertices. In PageRank, the score $R(w_i)$ of word w_i is defined as

$$R(w_i) = \lambda \sum_{j:w_j \rightarrow w_i} \frac{e(w_j, w_i)}{O(w_j)} R(w_j) + (1 - \lambda) \frac{1}{|V|}, \tag{2}$$

where λ is a *damping factor* range from 0 to 1, and $|V|$ is the number of vertices. The damping factor indicates that each vertex has a probability of $(1 - \lambda)$ to perform random jump to another vertex within this graph. PageRank scores are obtained by running Eq. (2) iteratively until convergence. The second term in Eq. (2) can be regarded as a smoothing factor to make the graph fulfill the property of being aperiodic and irreducible, so as to guarantee that PageRank converges to a unique stationary distribution. In PageRank, the second term is set to be the same value $\frac{1}{|V|}$ for all vertices within the graph, which indicates there are equal probabilities of random jump to all vertices.

In fact, the second term of PageRank in Eq. (2) can be set to be non-uniformed. Suppose we assign larger probabilities to some vertices, the final PageRank scores will prefer these vertices. We call this *Biased PageRank*.

The idea of Topical PageRank (TPR) is to run Biased PageRank for each topic separately. Each topic-specific PageRank prefers those words with high relevance to the corresponding topic. And the preferences are represented using random jump probabilities of words.

Formally, in the PageRank of a specific topic z, we will assign a topic-specific preference value $p_z(w)$ to each word w as its random jump probability with $\sum_{w \in V} p_z(w) = 1$. The words that are more relevant to topic z will be assigned larger probabilities when performing the PageRank. For topic z, the topic-specific PageRank scores are defined as follows:

$$R_z(w_i) = \lambda \sum_{j:w_j \to w_i} \frac{e(w_j, w_i)}{O(w_j)} R_z(w_j) + (1-\lambda) p_z(w_i). \tag{3}$$

In Fig. 1, we show an example with two topics. In this figure, we use the size of circles to indicate how relevant the word is to the topic. In the PageRanks of the two topics, high preference values will be assigned to different words with respect to the topic. Finally, the words will get different PageRank values in the two PageRanks.

The setting of preference values $p_z(w)$ will have a great influence to TPR. In this paper we use three measures to set preference values for TPR:

- $p_z(w) = pr(w|z)$, is the probability that word w occurs given topic z. This indicates how much that topic z focuses on word w.
- $p_z(w) = pr(z|w)$, is the probability of topic z given word w. This indicates how much that word w focuses on topic z.
- $p_z(w) = pr(w|z) \times pr(z|w)$, is the product of hub and authority values. This measure is inspired by the work in (Cohn and Chang, 2000).

Both PageRank and TPR are all iterative algorithms. We terminate the algorithms when the number of iterations reaches 100 or the difference of each vertex between two neighbor iterations is less than 0.001.

3.3 Extract Keyphrases Using Ranking Scores

After obtaining word ranking scores using TPR, we begin to rank candidate keyphrases. As reported in (Hulth, 2003), most manually assigned keyphrases turn out to be noun phrases. We thus select noun phrases from a document as candidate keyphrases for ranking.

The candidate keyphrases of a document is obtained as follows. The document is first tokenized. After that, we annotate the document with part-of-speech (POS) tags [1]. Third, we extract noun phrases with pattern `(adjective)*(noun)+`, which represents zero or more adjectives followed by one or more nouns. We regard these noun phrases as candidate keyphrases.

After identifying candidate keyphrases, we rank them using the ranking scores obtained by TPR. In PageRank for keyphrase extraction, the ranking score of a candidate keyphrase p is computed by summing up the ranking scores of all words within the phrase: $R(p) = \sum_{w_i \in p} R(w_i)$ (Mihalcea and Tarau, 2004; Wan and Xiao, 2008a; Wan and Xiao, 2008b). Then candidate keyphrases are ranked in descending order of ranking scores. The top M candidates are selected as keyphrases.

In TPR for keyphrase extraction, we first compute the ranking scores of candidate keyphrases separately for each topic. That is for each topic z we compute

$$R_z(p) = \sum_{w_i \in p} R_z(w_i). \tag{4}$$

By considering the topic distribution of document, We further integrate topic-specific rankings of candidate keyphrases into a final ranking and extract top-ranked ones as the keyphrases of the document. Denote the topic distribution of the document d as $pr(z|d)$ for each topic z. For each candidate keyphrase p, we compute its final ranking score as

[1] In experiments we use Stanford POS Tagger from `http://nlp.stanford.edu/software/tagger.shtml` with English tagging model `left3words-distsim-wsj`.

follows:

$$R(p) = \sum_{z=1}^{K} R_z(p) \times pr(z|d). \qquad (5)$$

After ranking candidate phrases in descending order of their integrated ranking scores, we select the top M as the keyphrases of document d.

4 Experiments

4.1 Datasets

To evaluate the performance of TPR for keyphrase extraction, we carry out experiments on two datasets.

One dataset was built by Wan and Xiao [2] which was used in (Wan and Xiao, 2008b). This dataset contains 308 news articles in DUC2001 (Over et al., 2001) with $2,488$ manually annotated keyphrases. There are at most 10 keyphrases for each document. In experiments we refer to this dataset as NEWS.

The other dataset was built by Hulth [3] which was used in (Hulth, 2003). This dataset contains $2,000$ abstracts of research articles and $19,254$ manually annotated keyphrases. In experiments we refer to this dataset as RESEARCH.

Since neither NEWS nor RESEARCH itself is large enough to learn efficient topics, we use the Wikipedia snapshot at March 2008 [4] to build topic interpreters with LDA. After removing non-article pages and the articles shorter than 100 words, we collected $2,122,618$ articles. After tokenization, stop word removal and word stemming, we build the vocabulary by selecting $20,000$ words according to their document frequency. We learn LDA models by taking each Wikipedia article as a document. In experiments we learned several models with different numbers of topics, from 50 to $1,500$ respectively. For the words absent in topic models, we simply set the topic distribution of the word as uniform distribution.

4.2 Evaluation Metrics

For evaluation, the words in both standard and extracted keyphrases are reduced to base forms using Porter Stemmer [5] for comparison. In experiments we select three evaluation metrics.

The first metric is precision/recall/F-measure represented as follows,

$$p = \frac{c_{correct}}{c_{extract}}, \quad r = \frac{c_{correct}}{c_{standard}}, \quad f = \frac{2pr}{p+r}, \qquad (6)$$

where $c_{correct}$ is the total number of correct keyphrases extracted by a method, $c_{extract}$ the total number of automatic extracted keyphrases, and $c_{standard}$ the total number of human-labeled standard keyphrases.

We note that the ranking order of extracted keyphrases also indicates the method performance. An extraction method will be better than another one if it can rank correct keyphrases higher. However, precision/recall/F-measure does not take the order of extracted keyphrases into account. To address the problem, we select the following two additional metrics.

One metric is *binary preference measure* (Bpref) (Buckley and Voorhees, 2004). Bpref is desirable to evaluate the performance considering the order in which the extracted keyphrases are ranked. For a document, if there are R correct keyphrases within M extracted keyphrases by a method, in which r is a correct keyphrase and n is an incorrect keyphrase, Bpref is defined as follows,

$$\text{Bpref} = \frac{1}{R} \sum_{r \in R} 1 - \frac{|n \text{ ranked higher than } r|}{M}. \qquad (7)$$

The other metric is *mean reciprocal rank* (MRR) (Voorhees, 2000) which is used to evaluate how the first correct keyphrase for each document is ranked. For a document d, $rank_d$ is denoted as the rank of the first correct keyphrase with all extracted keyphrases, MRR is defined as follows,

$$\text{MRR} = \frac{1}{|D|} \sum_{d \in D} \frac{1}{rank_d}, \qquad (8)$$

where D is the document set for keyphrase extraction.

Note that although the evaluation scores of most keyphrase extractors are still lower compared to

[2] http://wanxiaojun1979.googlepages.com.

[3] It was obtained from the author.

[4] http://en.wikipedia.org/wiki/Wikipedia_database.

[5] http://tartarus.org/~martin/PorterStemmer.

other NLP-tasks, it does not indicate the performance is poor because even different annotators may assign different keyphrases to the same document.

4.3 Influences of Parameters to TPR

There are four parameters in TPR that may influence the performance of keyphrase extraction including: (1) window size W for constructing word graph, (2) the number of topics K learned by LDA, (3) different settings of preference values $p_z(w)$, and (4) damping factor λ of TPR.

In this section, we look into the influences of these parameters to TPR for keyphrase extraction. Except the parameter under investigation, we set parameters to the following values: $W = 10$, $K = 1,000$, $\lambda = 0.3$ and $p_z(w) = pr(z|w)$, which are the settings when TPR achieves the best (or near best) performance on both `NEWS` and `RESEARCH`. In the following tables, we use "Pre.", "Rec." and "F." as the abbreviations of precision, recall and F-measure.

4.3.1 Window Size W

In experiments on `NEWS`, we find that the performance of TPR is stable when W ranges from 5 to 20 as shown in Table 1. This observation is consistent with the findings reported in (Wan and Xiao, 2008b).

Size	Pre.	Rec.	F.	Bpref	MRR
5	0.280	0.345	0.309	0.213	0.636
10	0.282	0.348	0.312	0.214	0.638
15	0.282	0.347	0.311	0.214	0.646
20	0.284	0.350	0.313	0.215	0.644

Table 1: Influence of window size W when the number of keyphrases $M = 10$ on `NEWS`.

Similarly, when W ranges from 2 to 10, the performance on `RESEARCH` does not change much. However, the performance on `NEWS` will become poor when $W = 20$. This is because the abstracts in `RESEARCH` (there are 121 words per abstract on average) are much shorter than the news articles in `NEWS` (there are 704 words per article on average). If the window size W is set too large on `RESEARCH`, the graph will become full-connected and the weights of links will tend to be equal, which cannot capture the local structure information of abstracts for keyphrase extraction.

4.3.2 The Number of Topics K

We demonstrate the influence of the number of topics K of LDA models in Table 2. Table 2 shows the results when K ranges from 50 to $1,500$ and $M = 10$ on `NEWS`. We observe that the performance does not change much as the number of topics varies until the number is much smaller ($K = 50$). The influence is similar on `RESEARCH` which indicates that LDA is appropriate for obtaining topics of words and documents for TPR to extract keyphrases.

K	Pre.	Rec.	F.	Bpref	MRR
50	0.268	0.330	0.296	0.204	0.632
100	0.276	0.340	0.304	0.208	0.632
500	0.284	0.350	0.313	0.215	0.648
1000	0.282	0.348	0.312	0.214	0.638
1500	0.282	0.348	0.311	0.214	0.631

Table 2: Influence of the number of topics K when the number of keyphrases $M = 10$ on `NEWS`.

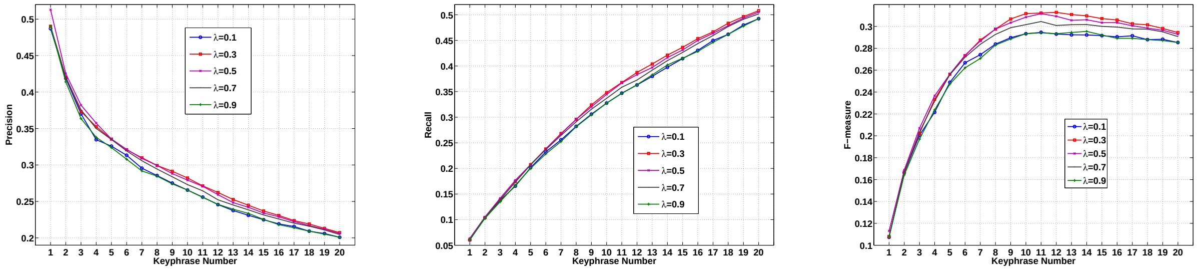
4.3.3 Damping Factor λ

Damping factor λ of TPR reconciles the influences of graph walks (the first term in Eq.(3)) and preference values (the second term in Eq.(3)) to the topic-specific PageRank scores. We demonstrate the influence of λ on `NEWS` in Fig. 2. This figure shows the precision/recall/F-measure when $\lambda = 0.1, 0.3, 0.5, 0.7, 0.9$ and M ranges from 1 to 20. From this figure we find that, when λ is set from 0.2 to 0.7, the performance is consistently good. The values of Bpref and MRR also keep stable with the variations of λ.

4.3.4 Preference Values

Finally, we explore the influences of different settings of preference values for TPR in Eq.(3). In Table 3 we show the influence when the number of keyphrases $M = 10$ on `NEWS`. From the table, we observe that $pr(z|w)$ performs the best. The similar observation is also got on `RESEARCH`.

In keyphrase extraction task, it is required to find the keyphrases that can appropriately represent the topics of the document. It thus does not want to extract those phrases that may appear in multiple topics like common words. The measure $pr(w|z)$ assigns preference values according to how frequently that words appear in the given topic. Therefore, the



(a) Precision

(b) Recall

(c) F-measure

Figure 2: Precision, recall and F-measure of TPR with $\lambda = 0.1, 0.3, 0.5, 0.7$ and 0.9 when M ranges from 1 to 20 on NEWS.

common words will always be assigned to a relatively large value in each topic-specific PageRank and finally obtain a high rank. $pr(w|z)$ is thus not a good setting of preference values in TPR. In the contrast, $pr(z|w)$ prefers those words that are focused on the given topic. Using $pr(z|w)$ to set preference values for TPR, we will tend to extract topic-focused phrases as keyphrases.

Pref	Pre.	Rec.	F.	Bpref	MRR
$pr(w\|z)$	0.256	0.316	0.283	0.192	0.584
$pr(z\|w)$	0.282	0.348	0.312	0.214	0.638
prod	0.259	0.320	0.286	0.193	0.587

Table 3: Influence of three preference value settings when the number of keyphrases $M = 10$ on NEWS.

4.4 Comparing with Baseline Methods

After we explore the influences of parameters to TPR, we obtain the best results on both NEWS and RESEARCH. We further select three baseline methods, i.e., TFIDF, PageRank and LDA, to compare with TPR.

The TFIDF computes the ranking scores of words based on words' $tfidf$ values in the document, namely $R(w) = tf_w \times \log(idf_w)$. While in PageRank (i.e., TextRank), the ranking scores of words are obtained using Eq.(2). The two baselines do not use topic information of either words or documents. The LDA computes the ranking score for each word using the topical similarity between the word and the document. Given the topics of the document d and a word w, We have used various methods to compute similarity including cosine similarity, predictive likelihood and KL-divergence (Heinrich, 2005), among which cosine similarity performs the best on both datasets. Therefore, we only show the results of the LDA baseline calculated using cosine similarity.

In Tables 4 and 5 we show the comparing results of the four methods on both NEWS and RESEARCH. Since the average number of manual-labeled keyphrases on NEWS is larger than RESEARCH, we set $M = 10$ for NEWS and $M = 5$ for RESEARCH. The parameter settings on both NEWS and RESEARCH have been stated in Section 4.3.

Method	Pre.	Rec.	F.	Bpref	MRR
TFIDF	0.239	0.295	0.264	0.179	0.576
PageRank	0.242	0.299	0.267	0.184	0.564
LDA	0.259	0.320	0.286	0.194	0.518
TPR	**0.282**	**0.348**	**0.312**	**0.214**	**0.638**

Table 4: Comparing results on NEWS when the number of keyphrases $M = 10$.

Method	Pre.	Rec.	F.	Bpref	MRR
TFIDF	0.333	0.173	0.227	0.255	0.565
PageRank	0.330	0.171	0.225	0.263	0.575
LDA	0.332	0.172	0.227	0.254	0.548
TPR	**0.354**	**0.183**	**0.242**	**0.274**	**0.583**

Table 5: Comparing results on RESEARCH when the number of keyphrases $M = 5$.

From the two tables, we have the following observations.

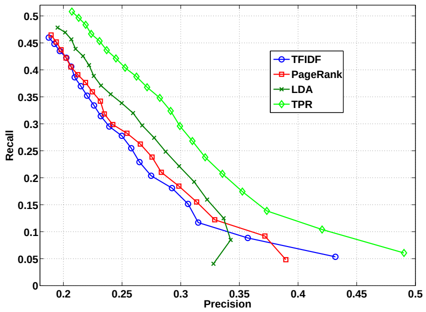


Figure 3: Precision-recall results on NEWS when M ranges from 1 to 20.

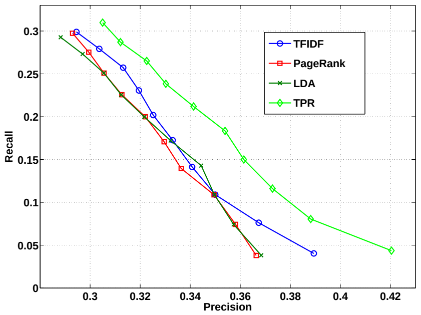


Figure 4: Precision-recall results on RESEARCH when M ranges from 1 to 10.

First, TPR outperform all baselines on both datasets. The improvements are all statistically significant tested with bootstrap re-sampling with 95% confidence. This indicates the robustness and effectiveness of TPR.

Second, LDA performs equal or better than TFIDF and PageRank under precision/recall/F-measure. However, the performance of LDA under MRR is much worse than TFIDF and PageRank, which indicates LDA fails to correctly extract the first keyphrase earlier than other methods. The reason is: (1) LDA does not consider the local structure information of document as PageRank, and (2) LDA also does not consider the frequency information of words within the document. In the contrast, TPR enjoys the advantages of both LDA and TFIDF/PageRank, by using the external topic information like LDA and internal document structure like TFIDF/PageRank.

Moreover, in Figures 3 and 4 we show the precision-recall relations of four methods on NEWS and RESEARCH. Each point on the precision-recall curve is evaluated on different numbers of extracted keyphrases M. The closer the curve to the upper right, the better the overall performance. The results again illustrate the superiority of TPR.

4.5 Extracting Example

At the end, in Table 6 we show an example of extracted keyphrases using TPR from a news article with title "Arafat Says U.S. Threatening to Kill PLO Officials" (The article number in DUC2001 is AP880510-0178). Here we only show the top 10 keyphrases, and the correctly extracted ones are marked with "(+)". We also mark the number of correctly extracted keyphrases after method name like "(+7)" after TPR. We also illustrate the top 3 topics of the document with their topic-specific keyphrases. It is obvious that the top topics, on "Palestine", "Israel" and "terrorism" separately, have a good coverage on the discussion objects of this article, which also demonstrate a good diversity with each other. By integrating these topic-specific keyphrases considering the proportions of these topics, we obtain the best performance of keyphrase extraction using TPR.

In Table 7 we also show the extracted keyphrases of baselines from the same news article. For TFIDF, it only considered the frequency properties of words, and thus highly ranked the phrases with "PLO" which appeared about 16 times in this article, and failed to extract the keyphrases on topic "Israel". LDA only measured the importance of words using document topics without considering the frequency information of words and thus missed keyphrases with high-frequency words. For example, LDA failed to extract keyphrase "political assassination", in which the word "assassination" occurred 8 times in this article.

5 Related Work

In this paper we proposed TPR for keyphrase extraction. A pioneering achievement in keyphrase extraction was carried out in (Turney, 1999) which regarded keyphrase extraction as a classification task. Generally, the supervised methods need manually annotated training set which is time-consuming and in this paper we focus on unsupervised method.

TPR (+7)
PLO leader Yasser Arafat(+), Abu Jihad, Khalil Wazir(+), slaying Wazir, political assassination(+), Palestinian guerrillas(+), particulary Palestinian circles, Israeli officials(+), Israeli squad(+), terrorist attacks(+)
TPR, Rank 1 Topic on “Palestine”
PLO leader Yasser Arafat(+), United States(+), State Department spokesman Charles Redman, Abu Jihad, U.S. government document, Palestine Liberation Organization leader, political assassination(+), Israeli officials(+), alleged document
TPR, Rank 2 Topic on “Israel”
PLO leader Yasser Arafat(+), United States(+), Palestine Liberation Organization leader, Israeli officials(+), U.S. government document, alleged document, Arab government, slaying Wazir, State Department spokesman Charles Redman, Khalil Wazir(+)
TPR, Rank 3 Topic on “terrorism”
terrorist attacks(+), PLO leader Yasser Arafat(+), Abu Jihad, United States(+), alleged document, U.S. government document, Palestine Liberation Organization leader, State Department spokesman Charles Redman, political assassination(+), full cooperation

Table 6: Extracted keyphrases by TPR.

TFIDF (+5)
PLO leader Yasser Arafat(+), PLO attacks, PLO offices, PLO officials(+), PLO leaders, Abu Jihad, terrorist attacks(+), Khalil Wazir(+), slaying wazir, political assassination(+)
PageRank (+3)
PLO leader Yasser Arafat(+), PLO officials(+), PLO attacks, United States(+), PLO offices, PLO leaders, State Department spokesman Charles Redman, U.S. government document, alleged document, Abu Jihad
LDA (+5)
PLO leader Yasser Arafat(+), Palestine Liberation Organization leader, Khalil Wazir(+), Palestinian guerrillas(+), Abu Jihad, Israeli officials(+), particulary Palestinian circles, Arab government, State Department spokesman Charles Redman, Israeli squad(+)

Table 7: Extracted keyphrases by baselines.

Starting with TextRank (Mihalcea and Tarau, 2004), graph-based ranking methods are becoming the most widely used unsupervised approach for keyphrase extraction. Litvak and Last (2008) applied HITS algorithm on the word graph of a document for keyphrase extraction. Although HITS itself worked the similar performance to PageRank, we plan to explore the integration of topics and HITS in future work. Wan (2008b; 2008a) used a small number of nearest neighbor documents to provide more knowledge for keyphrase extraction. Some methods used clustering techniques on word graphs for keyphrase extraction (Grineva et al., 2009; Liu et al., 2009). The clustering-based method performed well on short abstracts (with F-measure 0.382 on `RESEARCH`) but poorly on long articles (`NEWS` with F-measure score 0.216) due to two non-trivial issues: (1) how to determine the number of clusters, and (2) how to weight each cluster and select keyphrases from the clusters. In this paper we focus on improving graph-based methods via topic decomposition, we thus only compare with PageRank as well as TFIDF and LDA and do not compare with clustering-based methods in details.

In recent years, two algorithms were proposed to rank web pages by incorporating topic information of web pages within PageRank (Haveliwala, 2002; Nie et al., 2006). The method in (Haveliwala, 2002), is similar to TPR which also decompose PageRank into various topics. However, the method in (Haveliwala, 2002) only considered to set the preference values using $pr(w|z)$ (In the context of (Haveliwala, 2002), w indicates Web pages). In Section 4.3.4 we have shown that the setting of using $pr(z|w)$ is much better than $pr(w|z)$.

Nie et al. (2006) proposed a more complicated ranking method. In this method, topical PageRanks are performed together. The basic idea of (Nie et al., 2006) is, when surfing following a graph link from vertex w_i to w_j, the ranking score on topic z of w_i will have a higher probability to pass to the same topic of w_j and have a lower probability to pass to a different topic of w_j. When the inter-topic jump probability is 0, this method is identical to (Haveliwala, 2002). We implemented the method and found that the random jumps between topics did not help improve the performance for keyphrase extraction, and did not demonstrate the results of this method.

6 Conclusion and Future Work

In this paper we propose a new graph-based framework, Topical PageRank, which incorporates topic information within random walk for keyphrase extraction. Experiments on two datasets show that TPR achieves better performance than other baseline methods. We also investigate the influence of various parameters on TPR, which indicates the effectiveness and robustness of the new method.

We consider the following research directions as future work.

1. In this paper we obtained latent topics using LDA learned from Wikipedia. We design to obtain topics using other machine learning methods and from other knowledge bases, and investigate the influence to performance of keyphrase extraction.

2. In this paper we integrated topic information in PageRank. We plan to consider topic information in other graph-based ranking algorithms such as HITS (Kleinberg, 1999).

3. In this paper we used Wikipedia to train LDA by assuming Wikipedia is an extensive snapshot of human knowledge which can cover most topics talked about in NEWS and RESEARCH. In fact, the learned topics are highly dependent on the learning corpus. We will investigate the influence of corpus selection in training LDA for keyphrase extraction using TPR.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant No. 60873174. The authors would like to thank Anette Hulth and Xiaojun Wan for kindly sharing their datasets. The authors would also thank Xiance Si, Tom Chao Zhou, Peng Li for their insightful suggestions and comments.

References

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January.

C. Buckley and E.M. Voorhees. 2004. Retrieval evaluation with incomplete information. In *Proceedings of SIGIR*, pages 25–32.

David Cohn and Huan Chang. 2000. Learning to probabilistically identify authoritative documents. In *Proceedings of ICML*, pages 167–174.

M. Grineva, M. Grinev, and D. Lizorkin. 2009. Extracting key terms from noisy and multi-theme documents. In *Proceedings of WWW*, pages 661–670.

Taher H. Haveliwala. 2002. Topic-sensitive pagerank. In *Proceedings of WWW*, pages 517–526.

G. Heinrich. 2005. Parameter estimation for text analysis. *Web: http://www. arbylon. net/publications/text-est.*

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of SIGIR*, pages 50–57.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of EMNLP*, pages 216–223.

J.M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632.

T.K. Landauer, P.W. Foltz, and D. Laham. 1998. An introduction to latent semantic analysis. *Discourse Processes*, 25:259–284.

Marina Litvak and Mark Last. 2008. Graph-based keyword extraction for single-document summarization. In *Proceedings of the workshop Multi-source Multilingual Information Extraction and Summarization*, pages 17–24.

Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of EMNLP*, pages 257–266.

C.D. Manning and H. Schutze. 2000. *Foundations of statistical natural language processing*. MIT Press.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In *Proceedings of EMNLP*, pages 404–411.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. 1990. WordNet: An on-line lexical database. *International Journal of Lexicography*, 3:235–244.

Thuy Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *Proceedings of the 10th International Conference on Asian Digital Libraries*, pages 317–326.

Lan Nie, Brian D. Davison, and Xiaoguang Qi. 2006. Topical link analysis for web search. In *Proceedings of SIGIR*, pages 91–98.

P. Over, W. Liggett, H. Gilbert, A. Sakharov, and M. Thatcher. 2001. Introduction to duc-2001: An intrinsic evaluation of generic news text summarization systems. In *Proceedings of DUC2001*.

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, 1998*.

Peter D. Turney. 1999. Learning to extract keyphrases from text. *National Research Council Canada, Institute for Information Technology, Technical Report ERB-1057*.

Peter D. Turney. 2000. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2(4):303–336.

E.M. Voorhees. 2000. The trec-8 question answering track report. In *Proceedings of TREC*, pages 77–82.

Xiaojun Wan and Jianguo Xiao. 2008a. Collabrank: Towards a collaborative approach to single-document keyphrase extraction. In *Proceedings of COLING*, pages 969–976.

Xiaojun Wan and Jianguo Xiao. 2008b. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of AAAI*, pages 855–860.

Incorporating Content Structure into Text Analysis Applications

Christina Sauper, Aria Haghighi, Regina Barzilay
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{csauper, aria42, regina}@csail.mit.edu

Abstract

In this paper, we investigate how modeling content structure can benefit text analysis applications such as extractive summarization and sentiment analysis. This follows the linguistic intuition that rich contextual information should be useful in these tasks. We present a framework which combines a supervised text analysis application with the induction of latent content structure. Both of these elements are learned jointly using the EM algorithm. The induced content structure is learned from a large unannotated corpus and biased by the underlying text analysis task. We demonstrate that exploiting content structure yields significant improvements over approaches that rely only on local context.[1]

1 Introduction

In this paper, we demonstrate that leveraging document structure significantly benefits text analysis applications. As a motivating example, consider the excerpt from a DVD review shown in Table 1. This review discusses multiple aspects of a product, such as audio and video properties. While the word "pleased" is a strong indicator of positive sentiment, the sentence in which it appears does not specify the aspect to which it relates. Resolving this ambiguity requires information about global document structure.

A central challenge in utilizing such information lies in finding a relevant representation of content structure for a specific text analysis task. For instance, when performing single-aspect sentiment analysis, the most relevant aspect of content structure is whether a given sentence is objective or subjective (Pang and Lee, 2004). In a multi-aspect setting, however, information about the sentence topic is required to determine the aspect to which a sentiment-bearing word relates (Snyder and Barzilay, 2007). As we can see from even these closely related applications, the content structure representation should be intimately tied to a specific text analysis task.

> **Audio** Audio choices are English, Spanish and French Dolby Digital 5.1 ... Bass is still robust and powerful, giving weight to just about any scene – most notably the film's exciting final fight. Fans should be **pleased** with the presentation.
>
> **Extras** This single-disc DVD comes packed in a black amaray case with a glossy slipcover. Cover art has clearly been designed to appeal the Twilight crowd ... Finally, we've got a deleted scenes reel. Most of the excised scenes are actually pretty interesting.

Table 1: An excerpt from a DVD review.

In this work, we present an approach in which a content model is learned jointly with a text analysis task. We assume complete annotations for the task itself, but we learn the content model from raw, unannotated text. Our approach is implemented in a discriminative framework using latent variables to represent facets of content structure. In this framework, the original task features (e.g., lexical ones) are conjoined with latent variables to enrich the features with global contextual information. For example, in Table 1, the feature associated with the

[1] Code and processed data presented here are available at http://groups.csail.mit.edu/rbg/code/content_structure.html

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 377–387,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

word “pleased” should contribute most strongly to the sentiment of the *audio* aspect when it is augmented with a relevant topic indicator.

The coupling of the content model and the task-specific model allows the two components to mutually influence each other during learning. The content model leverages unannotated data to improve the performance of the task-specific model, while the task-specific model provides feedback to improve the relevance of the content model. The combined model can be learned effectively using a novel EM-based method for joint training.

We evaluate our approach on two complementary text analysis tasks. Our first task is a multi-aspect sentiment analysis task, where a system predicts the aspect-specific sentiment ratings (Snyder and Barzilay, 2007). Second, we consider a multi-aspect extractive summarization task in which a system extracts key properties for a pre-specified set of aspects. On both tasks, our method for incorporating content structure consistently outperforms structure-agnostic counterparts. Moreover, jointly learning content and task parameters yields additional gains over independently learned models.

2 Related Work

Prior research has demonstrated the usefulness of content models for discourse-level tasks. Examples of such tasks include sentence ordering (Barzilay and Lee, 2004; Elsner et al., 2007), extraction-based summarization (Haghighi and Vanderwende, 2009) and text segmentation (Chen et al., 2009). Since these tasks are inherently tied to document structure, a content model is essential to performing them successfully. In contrast, the applications considered in this paper are typically developed without any discourse information, focusing on capturing sentence-level relations. Our goal is to augment these models with document-level content information.

Several applications in information extraction and sentiment analysis are close in spirit to our work (Pang and Lee, 2004; Patwardhan and Riloff, 2007; McDonald et al., 2007). These approaches consider global contextual information when determining whether a given sentence is relevant to the underlying analysis task. All assume that relevant sentences have been annotated. For instance, Pang and Lee (2004) refine the accuracy of sentiment analysis by considering only the subjective sentences of a review as determined by an independent classifier. Patwardhan and Riloff (2007) take a similar approach in the context of information extraction. Rather than applying their extractor to all the sentences in a document, they limit it to event-relevant sentences. Since these sentences are more likely to contain information of interest, the extraction performance increases.

Another approach, taken by Choi and Cardie (2008) and Somasundaran et al. (2009) uses linguistic resources to create a latent model in a task-specific fashion to improve performance, rather than assuming sentence-level task relevancy. Choi and Cardie (2008) address a sentiment analysis task by using a heuristic decision process based on word-level intermediate variables to represent polarity. Somasundaran et al. (2009) similarly uses a bootstrapped local polarity classifier to identify sentence polarity.

McDonald et al. (2007) propose a model which jointly identifies global polarity as well as paragraph- and sentence-level polarity, all of which are observed in training data. While our approach uses a similar hierarchy, McDonald et al. (2007) is concerned with recovering the labels at all levels, whereas in this work we are interested in using latent document content structure as a means to benefit task predictions.

While our method also incorporates contextual information into existing text analysis applications, our approach is markedly different from the above approaches. First, our representation of context encodes more than the relevance-based binary distinction considered in the past work. Our algorithm adjusts the content model dynamically for a given task rather than pre-specifying it. Second, while previous work is fully supervised, in our case relevance annotations are readily available for only a few applications and are prohibitively expensive to obtain for many others. To overcome this drawback, our method induces a content model in an unsupervised fashion and connects it via latent variables to the target model. This design not only eliminates the need for additional annotations, but also allows the algorithm to leverage large quantities of raw data for training the content model. The tight coupling of relevance learning with the target analysis task leads to further performance gains.

Finally, our work relates to supervised topic models in Blei and McAullife (2007). In this work, latent topic variables are used to generate text as well as a supervised sentiment rating for the document. However, this architecture does not permit the usage of standard discriminative models which condition freely on textual features.

3 Model

3.1 Problem Formulation

In this section, we describe a model which incorporates content information into a multi-aspect summarization task.[2] Our approach assumes that at training time we have a collection of labeled documents $\mathcal{D}_L$, each consisting of the document text $\boldsymbol{s}$ and true task-specific labeling $\boldsymbol{y}^*$. For the multi-aspect summarization task, $\boldsymbol{y}^*$ consists of sequence labels (e.g., *value* or *service*) for the tokens of a document. Specifically, the document text $\boldsymbol{s}$ is composed of sentences $s_1, \ldots, s_n$ and the labelings $\boldsymbol{y}^*$ consists of corresponding label sequences $y_1, \ldots, y_n$.[3]

As is common in related work, we model each y_i using a CRF which conditions on the observed document text. In this work, we also assume a content model, which we fix to be the document-level HMM as used in Barzilay and Lee (2004). In this content model, each sentence s_i is associated with a hidden topic variable T_i which generates the words of the sentence. We will use $\boldsymbol{T} = (T_1, \ldots, T_n)$ to refer to the hidden topic sequence for a document. We fix the number of topics to a pre-specified constant K.

3.2 Model Overview

Our model, depicted in Figure 1, proceeds as follows: First the document-level HMM generates a hidden content topic sequence $\boldsymbol{T}$ for the sentences of a document. This content component is parametrized by θ and decomposes in the standard HMM fashion:[4]

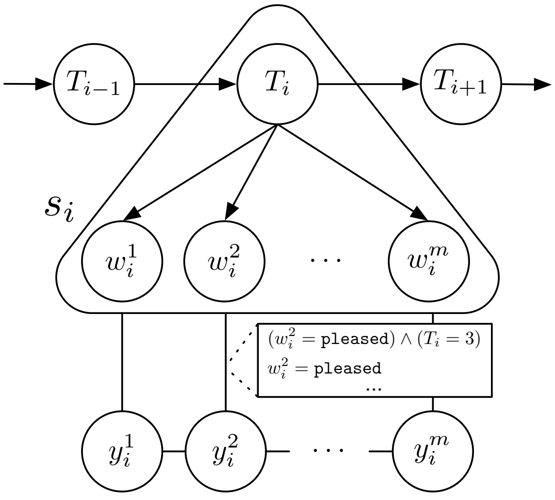


Figure 1: A graphical depiction of our model for sequence labeling tasks. The T_i variable represents the content model topic for the ith sentence s_i. The words of s_i, $(w_i^1, \ldots, w_i^m)$, each have a task label $(y_i^1, \ldots, y_i^m)$. Note that each token label has an undirected edge to a factor containing the words of the current sentence, s_i as well as the topic of the current sentence T_i.

$$P_\theta(\boldsymbol{s}, \boldsymbol{T}) = \prod_{i=1}^{n} P_\theta(T_i | T_{i-1}) \prod_{w \in s_i} P_\theta(w | T_i) \qquad (1)$$

Then the label sequences for each sentence in the document are independently modeled as CRFs which condition on both the sentence features and the sentence topic:

$$P_\phi(\boldsymbol{y} | \boldsymbol{s}, \boldsymbol{T}) = \prod_{i=1}^{n} P_\phi(y_i | s_i, T_i) \qquad (2)$$

Each sentence CRF is parametrized by ϕ and takes the standard form:

$$P_\phi(y|s, T) \propto \exp \left\{ \sum_j \phi^T \left[f_N(y^j, s, T) + f_E(y^j, y^{j+1}) \right] \right\}$$

[2] In Section 3.6, we discuss how this framework can be used for other text analysis applications.

[3] Note that each y_i is a label sequence across the words in s_i, rather than an individual label.

[4] We also utilize a hierarchical emission model so that each topic distribution interpolates between a topic-specific distribution as well as a shared background model; this is intended to capture domain-specific stop words.

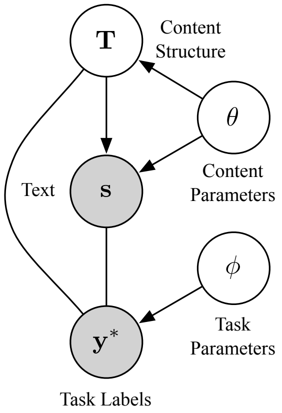


Figure 2: A graphical depiction of the generative process for a labeled document at training time (See Section 3); shaded nodes indicate variables which are observed at training time. First the latent underlying content structure $\boldsymbol{T}$ is drawn. Then, the document text $\boldsymbol{s}$ is drawn conditioned on the content structure utilizing content parameters θ. Finally, the observed task labels for the document are modeled given $\boldsymbol{s}$ and $\boldsymbol{T}$ using the task parameters ϕ. Note that the arrows for the task labels are undirected since they are modeled discriminatively.

where $f_N(\cdot)$ and $f_E(\cdot)$ are feature functions associated with CRF nodes and transitions respectively.

Allowing the CRF to condition on the sentence topic T_i permits predictions to be more sensitive to content. For instance, using the example from Table 1, we could have a feature that indicates the word "pleased" conjoined with the segment topic (see Figure 1). These topic-specific features serve to disambiguate word usage.

This joint process, depicted graphically in Figure 2, is summarized as:

$$P(\boldsymbol{T}, \boldsymbol{s}, \boldsymbol{y}^*) = P_\theta(\boldsymbol{T}, \boldsymbol{s}) P_\phi(\boldsymbol{y}^* | \boldsymbol{s}, \boldsymbol{T}) \qquad (3)$$

Note that this probability decomposes into a document-level HMM term (the content component) as well as a product of CRF terms (the task component).

3.3 Learning

During learning, we would like to find the document-level HMM parameters θ and the summarization task CRF parameters ϕ which maximize the likelihood of the labeled documents. The only observed elements of a labeled document are the document text $\boldsymbol{s}$ and the aspect labels $\boldsymbol{y}^*$. This objective is given by:

$$\begin{aligned}\mathcal{L}_L(\phi, \theta) &= \sum_{(\boldsymbol{s}, \boldsymbol{y}^*) \in \mathcal{D}_L} \log P(\boldsymbol{s}, \boldsymbol{y}^*) \\ &= \sum_{(\boldsymbol{s}, \boldsymbol{y}^*) \in \mathcal{D}_L} \log \sum_{\boldsymbol{T}} P(\boldsymbol{T}, \boldsymbol{s}, \boldsymbol{y}^*)\end{aligned}$$

We use the EM algorithm to optimize this objective.

E-Step The E-Step in EM requires computing the posterior distribution over latent variables. In this model, the only latent variables are the sentence topics $\boldsymbol{T}$. To compute this term, we utilize the decomposition in Equation (3) and rearrange HMM and CRF terms to obtain:

$$\begin{aligned}P(\boldsymbol{T}, \boldsymbol{s}, \boldsymbol{y}^*) &= P_\theta(\boldsymbol{T}, \boldsymbol{s}) P_\phi(\boldsymbol{y}^* | \boldsymbol{T}, \boldsymbol{s}) \\ &= \left(\prod_{i=1}^{n} P_\theta(T_i | T_{i-1}) \prod_{w \in s_i} P_\theta(w | T_i) \right) \cdot \\ &\quad \left(\prod_{i=1}^{n} P_\phi(y_i^* | s_i, T_i) \right) \\ &= \prod_{i=1}^{n} P_\theta(T_i | T_{i-1}) \cdot \\ &\quad \left(\prod_{w \in s_i} P_\theta(w | T_i) P_\phi(y_i^* | s_i, T_i) \right)\end{aligned}$$

We note that this expression takes the same form as the document-level HMM, except that in addition to emitting the words of a sentence, we also have an observation associated with the sentence sequence labeling. We treat each $P_\phi(y_i^* | s_i, T_i)$ as part of the node potential associated with the document-level HMM. We utilize the Forward-Backward algorithm as one would with the document-level HMM in isolation, except that each node potential incorporates this CRF term.

M-Step We perform separate M-Steps for content and task parameters. The M-Step for the content parameters is identical to the document-level HMM content model: topic emission and transition distributions are updated with expected counts derived from E-Step topic posteriors.

The M-Step for the task parameters does not have a closed-form solution. Recall that in the M-Step, we maximize the log probability of all random variables given expectations of latent variables. Using the decomposition in Equation (3), it is clear that the only component of the joint labeled document probability which relies upon the task parameters is $\log P_\phi(\boldsymbol{y}^*|\boldsymbol{s}, \boldsymbol{T})$. Thus for the M-Step, it is sufficient to optimize the following with respect to ϕ:

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{T}|\boldsymbol{s},\boldsymbol{y}^*} \log P_\phi(\boldsymbol{y}^*|\boldsymbol{s}, \boldsymbol{T}) \\
&= \sum_{i=1}^{n} \mathbb{E}_{T_i|s_i, y_i^*} \log P_\phi(y_i^*|s_i, T_i) \\
&= \sum_{i=1}^{n} \sum_{k=1}^{K} P(T_i = k|s_i, y_i^*) \log P_\phi(y_i^*|s_i, T_i)
\end{aligned}$$

The first equality follows from the decomposition of the task component into independent CRFs (see Equation (2)). Optimizing this objective is equivalent to a weighted version of the conditional likelihood objective used to train the CRF in isolation. An intuitive explanation of this process is that there are multiple CRF instances, one for each possible hidden topic T. Each utilizes different content features to explain the sentence sequence labeling. These instances are weighted according to the posterior over T obtained during the E-Step. While this objective is non-convex due to the summation over T, we can still optimize it using any gradient-based optimization solver; in our experiments, we used the LBFGS algorithm (Liu et al., 1989).

3.4 Inference

We must predict a label sequence y for each sentence s of the document. We assume a loss function over a sequence labeling y and a proposed labeling $\hat{y}$, which decomposes as:

$$L(y, \hat{y}) = \sum_j L(y^j, \hat{y}^j)$$

where each position loss is sensitive to the kind of error which is made. Failing to extract a token is penalized to a greater extent than extracting it with an incorrect label:

$$L(y^j, \hat{y}^j) = \begin{cases} 0 & \text{if } \hat{y}^j = y^j \\ c & \text{if } y^j \neq \text{NONE and } \hat{y}^j = \text{NONE} \\ 1 & \text{otherwise} \end{cases}$$

In this definition, NONE represents the background label which is reserved for tokens which do not correspond to labels of interest. The constant c represents a user-defined trade-off between precision and recall errors. For our multi-aspect summarization task, we select $c = 4$ for Yelp and $c = 5$ for Amazon to combat the high-precision bias typical of conditional likelihood models.

At inference time, we select the single labeling which minimizes the expected loss with respect to model posterior over label sequences:

$$\begin{aligned}
\hat{y} &= \min_{\hat{y}} \mathbb{E}_{y|\boldsymbol{s}} L(y, \hat{y}) \\
&= \min_{\hat{y}} \sum_{j=1} \mathbb{E}_{y^j|\boldsymbol{s}} L(y^j, \hat{y}^j)
\end{aligned}$$

In our case, we must marginalize out the sentence topic T:

$$\begin{aligned}
P(y^j|s) &= \sum_T P(y^j, T|s) \\
&= \sum_T P_\theta(T|s) P_\phi(y^j|s, T)
\end{aligned}$$

This minimum risk criterion has been widely used in NLP applications such as parsing (Goodman, 1999) and machine translation (DeNero et al., 2009). Note that the above formulation differs from the standard CRF due to the latent topic variables. Otherwise the inference task could be accomplished by directly obtaining posteriors over each y^j state using the Forward-Backwards algorithm on the sentence CRF.

Finding $\hat{y}$ can be done efficiently. First, we obtain marginal token posteriors as above. Then, the expected loss of a token prediction is computed as follows:

$$\sum_{\hat{y}^j} P(y^j|s) L(y^j, \hat{y}^j)$$

Once we obtain expected losses of each token prediction, we compute the minimum risk sequence labeling by running the Viterbi algorithm. The potential for each position and prediction is given by the negative expected loss. The maximal scoring sequence according to these potentials minimizes the expected risk.

3.5 Leveraging unannotated data

Our model allows us to incorporate unlabeled documents, denoted $\mathcal{D}_U$, to improve the learning of the content model. For an unlabeled document we only observe the document text s and assume it is drawn from the same content model as our labeled documents. The objective presented in Section 3.3 assumed that all documents were labeled; here we supplement this objective by capturing the likelihood of unlabeled documents according to the content model:

$$\begin{aligned}\mathcal{L}_U(\theta) &= \sum_{\boldsymbol{s}\in\mathcal{D}_U} \log P_\theta(\boldsymbol{s}) \\ &= \sum_{\boldsymbol{s}\in\mathcal{D}_U} \log \sum_{\boldsymbol{T}} P_\theta(\boldsymbol{s}, \boldsymbol{T})\end{aligned}$$

Our overall objective function is to maximize the likelihood of both our labeled and unlabeled data. This objective corresponds to:

$$\mathcal{L}(\phi, \theta) = \mathcal{L}_U(\theta) + \mathcal{L}_L(\phi, \theta)$$

This objective can also be optimized using the EM algorithm, where the E-Step for labeled and unlabeled documents is outlined above.

3.6 Generalization

The approach outlined can be applied to a wider range of task components. For instance, in Section 4.1 we apply this approach to multi-aspect sentiment analysis. In this task, the target y consists of numeric sentiment ratings $(y_1, \ldots, y_K)$ for each of K aspects. The task component consists of independent linear regression models for each aspect sentiment rating. For the content model, we associate a topic with each paragraph; $\boldsymbol{T}$ consists of assignments of topics to each document paragraph.

The model structure still decomposes as in Figure 2, but the details of learning are slightly different. For instance, because the task label (aspect sentiment ratings) is not localized to any region of the document, all content model variables influence the target response. Conditioned on the target label, all topic variables become correlated. Thus when learning, the E-Step requires computing a posterior over paragraph topic tuples $\boldsymbol{T}$:

$$P(\boldsymbol{T}|\boldsymbol{y}, \boldsymbol{s}) \propto P(\boldsymbol{s}, \boldsymbol{T})P(\boldsymbol{y}|\boldsymbol{T}, \boldsymbol{s})$$

For the case of our multi-aspect sentiment task, this computation can be done exactly by enumerating $\boldsymbol{T}$ tuples, since the number of sentences and possible topics is relatively small. If summation is intractable, the posterior may be approximated using variational techniques (Bishop, 2006), which is applicable to a broad range of potential applications.

4 Experimental Set-Up

We apply our approach to two text analysis tasks that stand to benefit from modeling content structure: multi-aspect sentiment analysis and multi-aspect review summarization.

4.1 Tasks

In the following section, we define each task in detail, explain the task-specific adaptation of the model and describe the data sets used in the experiments. Table 2 summarizes statistics for all the data sets.

For all tasks, when using a content model with a task model, we utilize a new set of features which include all the original features as well as a copy of each feature conjoined with the content topic assignment (see Figure 1). We also include a feature which indicates whether a given word was most likely emitted from the underlying topic or from a background distribution.

Multi-Aspect Sentiment Ranking The goal of multi-aspect sentiment classification is to predict a set of numeric ranks that reflects the user satisfaction for each aspect (Snyder and Barzilay, 2007). One of the challenges in this task is to attribute sentiment-bearing words to the aspects they describe. Information about document structure has the potential to greatly reduce this ambiguity.

Following standard sentiment ranking approaches (Wilson et al., 2004; Pang and Lee, 2005; Goldberg and Zhu, 2006; Snyder and Barzilay, 2007), we employ ordinary linear regression to independently map bag-of-words representations into predicted aspect ranks. In addition to commonly used lexical features, this set is augmented

Task	Labeled		Unlabeled	Avg. Size	
	Train	Test		Words	Sents
Multi-aspect sentiment	600	65	—	1,027	20.5
Multi-aspect summarization					
Amazon	35	24	12,684	214	11.7
Yelp	48	48	33,015	178	11.2

Table 2: This table summarizes the size of each corpus. In each case, the unlabeled texts of both labeled and unlabeled documents are used for training the content model, while only the labeled training corpus is used to train the task model. Note that the entire data set for the multi-aspect sentiment analysis task is labeled.

with content features as described above. For this application, we fix the number of HMM states to be equal to the predefined number of aspects.

We test our sentiment ranker on a set of DVD reviews from the website IGN.com.[5] Each review is accompanied by 1-10 scale ratings in four categories that assess the quality of a movie's content, video, audio, and DVD extras. In this data set, segments corresponding to each of the aspects are clearly delineated in each document. Therefore, we can compare the performance of the algorithm using automatically induced content models against the gold standard structural information.

Multi-Aspect Review Summarization The goal of this task is to extract informative phrases that identify information relevant to several predefined aspects of interest. In other words, we would like our system to both extract important phrases (e.g., *cheap food*) and label it with one of the given aspects (e.g., *value*). For concrete examples and lists of aspects for each data set, see Figures 3b and 3c. Variants of this task have been considered in review summarization in previous work (Kim and Hovy, 2006; Branavan et al., 2009).

This task has elements of both information extraction and phrase-based summarization — the phrases we wish to extract are broader in scope than in standard template-driven IE, but at the same time, the type of selected information is restricted to the defined aspects, similar to query-based summarization. The difficulty here is that phrase selection is highly context-dependent. For instance, in TV reviews such as in Figure 3b, the highlighted phrase "easy to read" might refer to either the menu or the remote; broader context is required for correct labeling.

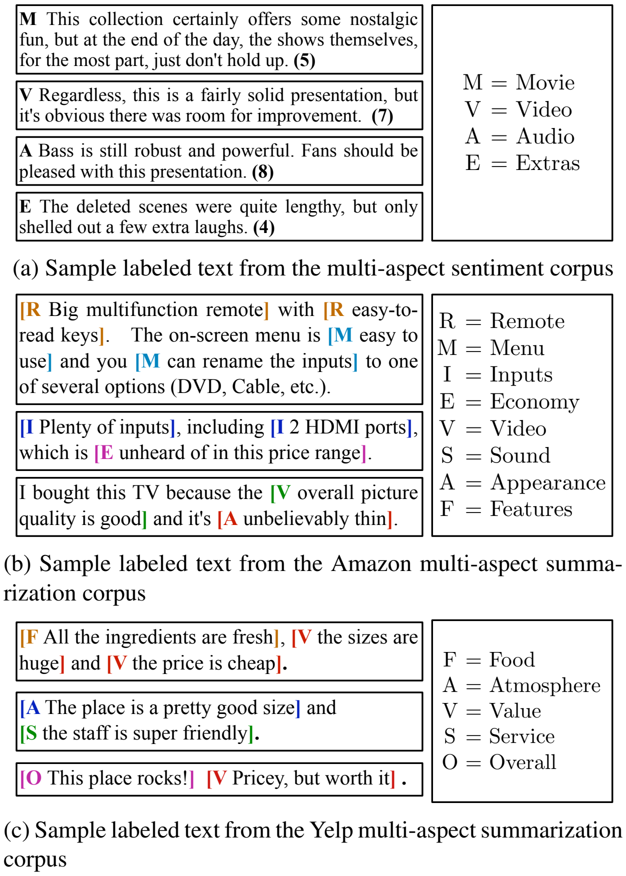
We evaluated our approach for this task on two data sets: Amazon TV reviews (Figure 3b) and Yelp restaurant reviews (Figure 3c). To eliminate noisy reviews, we only retain documents that have been rated "helpful" by the users of the site; we also remove reviews which are abnormally short or long.

Each data set was manually annotated with aspect labels using Mechanical Turk, which has been used in previous work to annotate NLP data (Snow et al., 2008). Since we cannot select high-quality annotators directly, we included a control document which had been previously annotated by a native speaker among the documents assigned to each annotator. The work of any annotator who exhibited low agreement on the control document annotation was excluded from the corpus. To test task annotation agreement, we use Cohen's Kappa (Cohen, 1960). On the Amazon data set, two native speakers annotated a set of four documents. The agreement between the judges was 0.54. On the Yelp data set, we simply computed the agreement between all pairs of reviewers who received the same control documents; the agreement was 0.49.

4.2 Baseline Comparison and Evaluation

Baselines For all the models, we obtain a baseline system by eliminating content features and only using a task model with the set of features described above. We also compare against a simplified variant of our method wherein a content model is induced in isolation rather than learned jointly in the context of the underlying task. In our experiments, we refer to the two methods as the No Content Model (NoCM) and Independent Content Model (IndepCM) settings, respectively. The Joint Content

[5] http://dvd.ign.com/index/reviews.html



M This collection certainly offers some nostalgic fun, but at the end of the day, the shows themselves, for the most part, just don't hold up. **(5)**

V Regardless, this is a fairly solid presentation, but it's obvious there was room for improvement. **(7)**

A Bass is still robust and powerful. Fans should be pleased with this presentation. **(8)**

E The deleted scenes were quite lengthy, but only shelled out a few extra laughs. **(4)**

M = Movie
V = Video
A = Audio
E = Extras

(a) Sample labeled text from the multi-aspect sentiment corpus

[R Big multifunction remote] with [R easy-to-read keys]. The on-screen menu is [M easy to use] and you [M can rename the inputs] to one of several options (DVD, Cable, etc.).

[I Plenty of inputs], including [I 2 HDMI ports], which is [E unheard of in this price range].

I bought this TV because the [V overall picture quality is good] and it's [A unbelievably thin].

R = Remote
M = Menu
I = Inputs
E = Economy
V = Video
S = Sound
A = Appearance
F = Features

(b) Sample labeled text from the Amazon multi-aspect summarization corpus

[F All the ingredients are fresh], [V the sizes are huge] and [V the price is cheap].

[A The place is a pretty good size] and [S the staff is super friendly].

[O This place rocks!] [V Pricey, but worth it] .

F = Food
A = Atmosphere
V = Value
S = Service
O = Overall

(c) Sample labeled text from the Yelp multi-aspect summarization corpus

Figure 3: Excerpts from the three corpora with the corresponding labels. Note that sentences from the multi-aspect summarization corpora generally focus on only one or two aspects. The multi-aspect sentiment corpus has labels per paragraph rather than per sentence.

Model (JointCM) setting refers to our full model described in Section 3, where content and task components are learned jointly.

Evaluation Metrics For multi-aspect sentiment ranking, we report the average L_2 (squared difference) and L_1 (absolute difference) between system prediction and true 1-10 sentiment rating across test documents and aspects.

For the multi-aspect summarization task, we measure average token precision and recall of the label assignments (Multi-label). For the Amazon corpus, we also report a coarser metric which measures extraction precision and recall while ignoring labels (Binary labels) as well as ROUGE (Lin, 2004). To compute ROUGE, we control for length by limiting each system to predict the same number of tokens as the original labeled document.

	L_1	L_2
NoCM	1.37	3.15
IndepCM	1.28†*	2.80†*
JointCM	**1.25**†	**2.65**†*
Gold	1.18†*	2.48†*

Table 3: The error rate on the multi-aspect sentiment ranking. We report mean L_1 and L_2 between system prediction and true values over all aspects. Marked results are statistically significant with $p < 0.05$: * over the previous model and † over NoCM.

	F_1	F_2	Prec.	Recall
NoCM	28.8%	34.8%	22.4%	40.3%
IndepCM	37.9%	43.7%	31.1%†*	**48.6%**†*
JointCM	**39.2%**	**44.4%**	**32.9%**†*	**48.6%**†

Table 4: Results for multi-aspect summarization on the Yelp corpus. Marked precision and recall are statistically significant with $p < 0.05$: * over the previous model and † over NoCM.

Our metrics of statistical significance vary by task. For the sentiment task, we use Student's t-test. For the multi-aspect summarization task, we perform chi-square analysis on the ROUGE scores as well as on precision and recall separately, as is commonly done in information extraction (Freitag, 2004; Weeds et al., 2004; Finkel and Manning, 2009).

5 Results

In this section, we present the results of the methods on the tasks described above (see Tables 3, 4, and 5).

Baseline Comparisons Adding a content model significantly outperforms the NoCM baseline on both tasks. The highest F_1 error reduction – 14.7% – is achieved on multi-aspect summarization on the Yelp corpus, followed by the reduction of 11.5% and 8.75%, on multi-aspect summarization on the Amazon corpus and multi-aspect sentiment ranking, respectively.

We also observe a consistent performance boost when comparing against the IndepCM baseline. This result confirms our hypothesis about the ad-

	Multi-label				Binary labels				
	F_1	F_2	Prec.	Recall	F_1	F_2	Prec.	Recall	ROUGE
NoCM	18.9%	18.0%	20.4%	17.5%	35.1%	33.6%	38.1%	32.6%	43.8%
IndepCM	24.5%	23.8%	**25.8%**†*	23.3%†*	43.0%	41.8%	**45.3%**†*	40.9%†*	47.4%†*
JointCM	**28.2%**	**31.3%**	24.3%†	**33.7%**†*	**47.8%**	**53.0%**	41.2%†	**57.1%**†*	**47.6%**†*

Table 5: Results for multi-aspect summarization on the Amazon corpus. Marked ROUGE, precision, and recall are statistically significant with $p < 0.05$: * over the previous model and † over NoCM.

vantages of jointly learning the content model in the context of the underlying task.

Comparison with additional context features One alternative to an explicit content model is to simply incorporate additional features into NoCM as a proxy for contextual information. In the multi-aspect summarization case, this can be accomplished by adding unigram features from the sentences before and after the current one.[6]

When testing this approach, however, the performance of NoCM actually decreases on both Amazon (to 15.0% F_1) and Yelp (to 24.5% F_1) corpora. This result is not surprising for this particular task – by adding these features, we substantially increase the feature space without increasing the amount of training data. An advantage of our approach is that our learned representation of context is coarse, and we can leverage large quantities of unannotated training data.

Impact of content model quality on task performance In the multi-aspect sentiment ranking task, we have access to gold standard document-level content structure annotation. This affords us the ability to compare the ideal content structure, provided by the document authors, with one that is learned automatically. As Table 3 shows, the manually created document structure segmentation yields the best results. However, the performance of our JointCM model is not far behind the gold standard content structure.

The quality of the induced content model is determined by the amount of training data. As Figure 4 shows, the multi-aspect summarizer improves with the increase in the size of raw data available for learning content model.

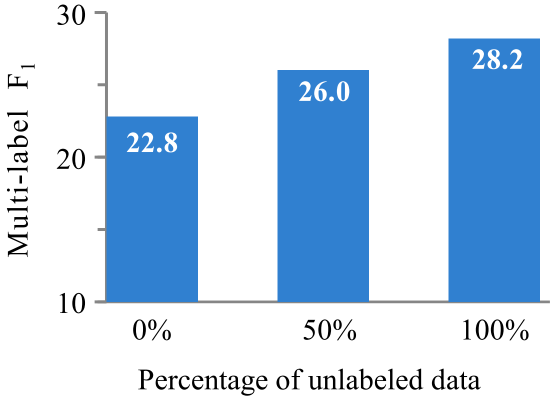


Figure 4: Results on the Amazon corpus using the complete annotated set with varying amounts of additional unlabeled data.[7]

Compensating for annotation sparsity We hypothesize that by incorporating rich contextual information, we can reduce the need for manual task annotation. We test this by reducing the amount of annotated data available to the model and measuring performance at several quantities of unannotated data. As Figure 5 shows, the performance increase achieved by doubling the amount of annotated data can also be achieved by adding only 12.5% of the unlabeled data.

6 Conclusion

In this paper, we demonstrate the benefits of incorporating content models in text analysis tasks. We also introduce a framework to allow the joint learning of an unsupervised latent content model with a supervised task-specific model. On multiple tasks and datasets, our results empirically connect model quality and task performance, suggesting that fur-

[6] This type of feature is not applicable to our multi-aspect sentiment ranking task, as we already use unigram features from the entire document.

[7] Because we append the unlabeled versions of the labeled data to the unlabeled set, even with 0% additional unlabeled data, there is a small data set to train the content model.

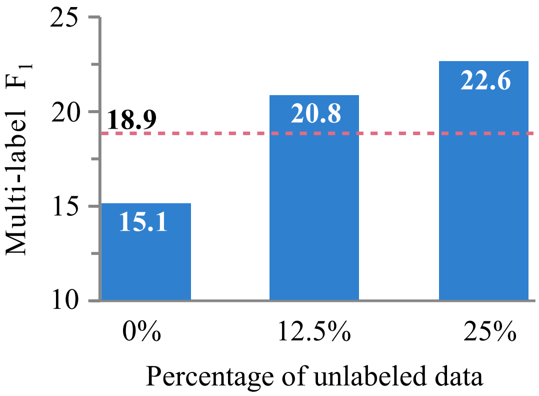


Figure 5: Results on the Amazon corpus using half of the annotated training documents. The content model is trained with 0%, 12.5%, and 25% of additional unlabeled data.[7] The dashed horizontal line represents NoCM with the complete annotated set.

ther improvements in content modeling may yield even further gains.

Acknowledgments

The authors acknowledge the support of the NSF (CAREER grant IIS-0448168) and NIH (grant 5-R01-LM009723-02). Thanks to Peter Szolovits and the MIT NLP group for their helpful comments. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of the funding organizations.

References

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *Proceedings of the NAACL/HLT*, pages 113–120.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc.

David M. Blei and Jon D. McAullife. 2007. Supervised Topic Models. In *NIPS*.

S. R. K. Branavan, Harr Chen, Jacob Eisenstein, and Regina Barzilay. 2009. Learning document-level semantic properties from free-text annotations. *JAIR*, 34:569–603.

Harr Chen, S. R. K. Branavan, Regina Barzilay, and David R. Karger. 2009. Content modeling using latent permutations. *JAIR*, 36:129–163.

Yejin Choi and Claire Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of the EMNLP*, pages 793–801.

J. Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37.

John DeNero, David Chiang, and Kevin Knight. 2009. Fast consensus decoding over translation forests. In *Proceedings of the ACL/IJCNLP*, pages 567–575.

Micha Elsner, Joseph Austerweil, and Eugene Charniak. 2007. A unified local and global model for discourse coherence. In *Proceedings of the NAACL/HLT*, pages 436–443.

Jenny Rose Finkel and Christopher D. Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of the NAACL*.

Dayne Freitag. 2004. Trained named entity recognition using distributional clusters. In *Proceedings of the EMNLP*, pages 262–269.

Andrew B. Goldberg and Xiaojin Zhu. 2006. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *Proceedings of the NAACL/HLT Workshop on TextGraphs*, pages 45–52.

Joshua Goodman. 1999. Semiring parsing. *Computational Linguistics*, 25(4):573–605.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of the NAACL/HLT*, pages 362–370.

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.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of the ACL*, pages 74–81.

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

Ryan McDonald, Kerry Hannan, Tyler Neylon, Mike Wells, and Jeff Reynar. 2007. Structured models for fine-to-coarse sentiment analysis. In *Proceedings of the ACL*, pages 432–439.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*, pages 271–278.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*, pages 115–124.

Siddharth Patwardhan and Ellen Riloff. 2007. Effective information extraction with semantic affinity patterns and relevant regions. In *Proceedings of the EMNLP/CoNLL*, pages 717–727.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast - but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the EMNLP*.

Benjamin Snyder and Regina Barzilay. 2007. Multiple aspect ranking using the good grief algorithm. In *Proceedings of the NAACL/HLT*, pages 300–307.

Swapna Somasundaran, Galileo Namata, Janyce Wiebe, and Lise Getoor. 2009. Supervised and unsupervised methods in employing discourse relations for improving opinion polarity classification. In *Proceedings of the EMNLP*, pages 170–179.

Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of the COLING*, page 1015.

Theresa Wilson, Janyce Wiebe, and Rebecca Hwa. 2004. Just how mad are you? finding strong and weak opinion clauses. In *Proceedings of the AAAI*, pages 761–769.

Exploiting Conversation Structure in Unsupervised Topic Segmentation for Emails

Shafiq Joty and Giuseppe Carenini and Gabriel Murray and Raymond T. Ng
{rjoty, carenini, gabrielm, rng}@cs.ubc.ca
Department of Computer Science
University of British Columbia
Vancouver, BC, V6T 1Z4, Canada

Abstract

This work concerns automatic topic segmentation of email conversations. We present a corpus of email threads manually annotated with topics, and evaluate annotator reliability. To our knowledge, this is the first such email corpus. We show how the existing topic segmentation models (i.e., Lexical Chain Segmenter (LCSeg) and Latent Dirichlet Allocation (LDA)) which are solely based on lexical information, can be applied to emails. By pointing out where these methods fail and what any desired model should consider, we propose two novel extensions of the models that not only use lexical information but also exploit finer level conversation structure in a principled way. Empirical evaluation shows that LCSeg is a better model than LDA for segmenting an email thread into topical clusters and incorporating conversation structure into these models improves the performance significantly.

1 Introduction

With the ever increasing popularity of emails and web technologies, it is very common for people to discuss issues, events, agendas or tasks by email. Effective processing of the email contents can be of great strategic value. In this paper, we study the problem of *topic segmentation for emails*, i.e., grouping the sentences of an email thread into a set of coherent topical clusters. Adapting the standard definition of topic (Galley et al., 2003) to conversations/emails, we consider a topic is something about which the participant(s) discuss or argue or express their opinions. For example, in the email thread shown in Figure 1, according to the majority of our annotators, participants discuss three topics (e.g., 'telecon cancellation', 'TAG document', and 'responding to I18N'). Multiple topics seem to occur naturally in social interactions, whether synchronous (e.g., chats, meetings) or asynchronous (e.g., emails, blogs) conversations. In multi-party chat (Elsner and Charniak, 2008) report an average of 2.75 discussions active at a time. In our email corpus, we found an average of 2.5 topics per thread.

Topic segmentation is often considered a prerequisite for other higher-level conversation analysis and applications of the extracted structure are broad, encompassing: summarization (Harabagiu and Lacatusu, 2005), information extraction and ordering (Allan, 2002), information retrieval (Dias et al., 2007), and intelligent user interfaces (Dredze et al., 2008). While extensive research has been conducted in topic segmentation for monologues (e.g., (Malioutov and Barzilay, 2006), (Choi et al., 2001)) and synchronous dialogs (e.g., (Galley et al., 2003), (Hsueh et al., 2006)), none has studied the problem of segmenting asynchronous multi-party conversations (e.g., email). Therefore, there is no reliable annotation scheme, no standard corpus, and no agreed-upon metrics available. Also, it is our key hypothesis that, because of its asynchronous nature, and the use of quotation (Crystal, 2001), topics in an email thread often do not change in a sequential way. As a result, we do not expect models which have proved successful in monologue or dialog to be as effective when they are applied to email conversations.

Our contributions in this paper aim to remedy

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 388–398,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

these problems. First, we present an email corpus annotated with topics and evaluate annotator agreement. Second, we adopt a set of metrics to measure the local and global structural similarity between two annotations from the work on multi-party chat disentanglement (Elsner and Charniak, 2008). Third, we show how the two state-of-the-art topic segmentation methods (i.e., LCSeg and LDA) which are solely based on lexical information and make strong assumptions on the resulting topic models, can be effectively applied to emails, by having them to consider, in a principled way, a finer level structure of the underlying conversations. Experimental results show that both LCSeg and LDA benefit when they are extended to consider the conversational structure. When comparing the two methods, we found that LCSeg is better than LDA and this advantage is preserved when they are extended to incorporate conversational structure.

2 Related Work

Three research areas are directly related to our study: a) text segmentation models, b) probabilistic topic models, and c) extracting and representing the conversation structure of emails.

Topic segmentation has been extensively studied both for monologues and dialogs. (Malioutov and Barzilay, 2006) uses the minimum cut model to segment spoken lectures (i.e., monologue). They form a weighted undirected graph where the vertices represent sentences and the weighted links represent the similarity between sentences. Then the segmentation problem can be solved as a graph partitioning problem, where the assumption is that the sentences in a segment should be similar, while sentences in different segments should be dissimilar. They optimize the 'normalized cut' criterion to extract the segments. In general, the minimization of the normalized cut criterion is NP-complete. However, the linearity constraint on text segmentation for monologue allows them to find an exact solution in polynomial time. In our extension of LCSeg, we use a similar method to consolidate different segments; however, in our case the linearity constraint is absent. Therefore, we approximate the optimal solution by spectral clustering (Shi and Malik, 2000). Moving to the task of segmenting dialogs, (Galley et al., 2003) first proposed the lexical chain based unsupervised segmenter (LCSeg) and a supervised segmenter for segmenting meeting transcripts. Their supervised approach uses C4.5 and C4.5 rules binary classifiers with lexical and conversational features (e.g., cue phrase, overlap, speaker, silence, and lexical cohesion function). Their supervised approach performs significantly better than LCSeg. (Hsueh et al., 2006) follow the same approaches as (Galley et al., 2003) on both manual transcripts and ASR output of meetings. They perform segmentation at both coarse (topic) and fine (subtopic) levels. For the topic level, they achieve similar results as (Galley et al., 2003), with the supervised approach outperforming LCSeg. However, for the subtopic level, LCSeg performs significantly better than the supervised one. In our work, we show how LCSeg performs when applied to the temporal ordering of the emails in a thread. We also propose its extension to leverage the finer conversation structure of emails.

The probabilistic generative topic models, such as LDA and its variants (e.g., (Blei et al., 2003), (Steyvers and Griffiths, 2007)), have proven to be successful for topic segmentation in both monologue (e.g., (Chen et al., 2009)) and dialog (e.g., (Georgescul et al., 2008)). (Purver et al., 2006) uses a variant of LDA for the tasks of segmenting meeting transcripts and extracting the associated topic labels. However, their approach for segmentation does not perform better than LCSeg. In our work, we show how the general LDA performs when applied to email conversations and describe how it can be extended to exploit the conversation structure of emails.

Several approaches have been proposed to capture an email conversation . Email programs (e.g., Gmail, Yahoomail) group emails into threads using headers. However, our annotations show that topics change at a finer level of granularity than emails. (Carenini et al., 2007) present a method to capture an email conversation at the finer level by analyzing the embedded quotations in emails. A fragment quotation graph (FQG) is generated, which is shown to be beneficial for email summarization. In this paper, we show that topic segmentation models can also benefit significantly from this fine conversation structure of email threads.

3 Corpus and Evaluation Metrics

There are no publicly available email corpora annotated with topics. Therefore, the first step was to develop our own corpus. We have annotated the BC3 email corpus (Ulrich et al., 2008) with topics[1]. The BC3 corpus, previously annotated with sentence level speech acts, meta sentence, subjectivity, extractive and abstractive summaries, is one of a growing number of corpora being used for email research. The corpus contains 40 email threads from the W3C corpus[2]. It has 3222 sentences and an average of 5 emails per thread.

3.1 Topic Annotation

Topic segmentation in general is a nontrivial and subjective task (Hsueh et al., 2006). The conversation phenomenon called 'Schism' makes it even more challenging for conversations. In schism a new conversation takes birth from an existing one, not necessarily because of a topic shift but because some participants refocus their attention onto each other, and away from whoever held the floor in the parent conversation and the annotators can disagree on the birth of a new topic (Aoki et al., 2006). In the example email thread shown in Figure 1, a schism takes place when people discuss about 'responding to I18N'. All the annotators do not agree on the fact that the topic about 'responding to I18N' swerves from the one about 'TAG document'. The annotators can disagree on the number of topics (i.e., some are specific and some are general), and on the topic assignment of the sentences[3]. To properly design an effective annotation manual and procedure we performed a two-phase pilot study before carrying out the actual annotation. For the pilot study we picked five email threads randomly from the corpus. In the first phase of the pilot study we selected five university graduate students to do the annotation. We then revised our instruction manual based on their feedback and the source of disagreement found. In the second phase we tested with a university postdoc doing the annotation.

For the actual annotation we selected three computer science graduates who are also native speakers of English. They annotated 39 threads of the BC3 corpus[4]. On an average they took seven hours to annotate the whole dataset.

BC3 contains three human written abstract summaries for each email thread. With each email thread the annotators were also given an associated human written summary to give a brief overview of the corresponding conversation. The task of finding topics was carried out in two phases. In the first phase, the annotators read the conversation and the associated summary and list the topics discussed. They specify the topics by a short description (e.g., "meeting agenda", "location and schedule") which provides a high-level overview of the topic. The target number of topics and the topic labels were not given in advance and they were instructed to find as many topics as needed to convey the overall content structure of the conversation.

In the second phase the annotators identify the most appropriate topic for each sentence. However, if a sentence covers more than one topic, they were asked to label it with all the relevant topics according to their order of relevance. If they find any sentence that does not fit into any topic, they are told to label those as the predefined topic 'OFF-TOPIC'. Wherever appropriate they were also asked to make use of two other predefined topics: 'INTRO' and 'END'. INTRO (e.g., 'hi', 'hello') signifies the section (usually at the beginning) of an email that people use to begin their email. Likewise, END (e.g., 'Cheers', 'Best') signifies the section (usually at the end) that people use to end their email. The annotators carried out the task on paper. We created the hierarchical thread view ('reply to' relation) using 'TAB's (indentation) and each participant's name is printed in a different color as in Gmail.

Table 1 shows some basic statistics computed on the three annotations of the 39 email threads[5]. On

[1] The BC3 corpus had already been annotated for email summarization, speech act recognition and subjectivity detection. This new annotation with topics will be also made publicly available at http://www.cs.ubc.ca/labs/lci/bc3.html

[2] http://research.microsoft.com/en-us/um/people/nickcr/w3c-summary.html

[3] The annotators also disagree on the topic labels, however in this work we are not interested in finding the topic labels.

[4] The annotators in the pilot and in the actual study were different so we could reuse the threads used in pilot study. However, one thread on which the pilot annotators agree fully, was used as an example in the instruction manual. This gives 39 threads left for the actual study.

[5] We got 100% agreement on the two predefined topics 'IN-

average we have 26.3 sentences and 2.5 topics per thread. A topic contains an average of 12.6 sentences. The average number of topics active at a time is 1.4. The average entropy is 0.94 and corresponds (as described in detail in the next section) to the granularity of the annotation. These statistics (number of topics and topic density) indicate that the dataset is suitable for topic segmentation.

	Mean	Max	Min
Number of sentences	26.3	55	13
Number of topics	2.5	7	1
Avg. topic length	12.6	35	3
Avg. topic density	1.4	3.1	1
Entropy	0.94	2.7	0

Table 1: Corpus statistics of human annotations

Metrics	Mean	Max	Min
1-to-1	0.804	1	0.31
loc_k	0.831	1	0.43
m-to-1	0.949	1	0.61

Table 2: Annotator agreement in the scale of 0 to 1

3.2 Evaluation Metrics

In this section we describe the metrics used to compare different human annotations and system's output. As different annotations (or system's output) can group sentences in different number of clusters, metrics widely used in classification, such as the κ statistic, are not applicable. Again, our problem of topic segmentation for emails is not sequential in nature. Therefore, the standard metrics widely used in sequential topic segmentation for monologues and dialogs, such as P_k and $WindowDiff(WD)$, are also not applicable. We adopt the more appropriate metrics 1-to-1, loc_k and m-to-1, introduced recently by (Elsner and Charniak, 2008). The 1-to-1 metric measures the global similarity between two annotations. It pairs up the clusters from the two annotations in a way that maximizes (globally) the total overlap and then reports the percentage of overlap. loc_k measures the local agreement within a context of k sentences. To compute the loc_3 metric for the m-th sentence in the two annotations, we consider the previous 3 sentences: m-1, m-2 and m-3, and mark them as either 'same' or 'different' depending on their topic assignment. The loc_3 score between two annotations is the mean agreement on these 'same' or 'different' judgments, averaged over all sentences. We report the agreement found in 1-to-1 and loc_k in Table 2. In both of the metrics we get high agreement, though the local agreement (average of 83%) is little higher than the global agreement (average of 80%).

If we consider the topic of a randomly picked sentence as a random variable then its entropy measures the level of detail in an annotation. If the topics are evenly distributed then the uncertainty (i.e., entropy) is higher. It also increases with the increase of the number of topics. Therefore, it is a measure of how specific an annotator is and in our dataset it varies from 0 [6] to 2.7. To measure how much the annotators agree on the general structure we use the m-to-1 metric. It maps each of the source clusters to the single target cluster with which it gets the highest overlap, then computes the total percentage of overlap. This metric is asymmetrical and not a measure to be optimized[7], but it gives us some intuition about specificity (Elsner and Charniak, 2008). If one annotator divides a cluster into two clusters then, the m-to-1 metric from fine to coarse is 1. In our corpus by mapping from fine to coarse we get an m-to-1 average of 0.949.

4 Topic Segmentation Models

Developing automatic tools for segmenting an email thread is challenging. The example email thread in Figure 1 demonstrates why. We use different colors and fonts to represent sentences of different topics[8]. One can notice that email conversations are different from written monologues (e.g., newspaper) and dialogs (e.g., meeting, chat) in various ways. As a communication media Email is distributed (unlike face to face meeting) and asynchronous (unlike

TRO' and 'END'. In all our computation (i.e., statistics, agreement, system's input) we excluded the sentences marked as either 'INTRO' or 'END'

[6] 0 uncertainty happens when there is only one topic found

[7] hence we do not use it to compare our models.

[8] 2 of the 3 annotators agree on this segmentation. Green represents topic 1 ('telecon cancellation'), orange indicates topic 2 ('TAG document') and magenta represents topic 3 ('responding to I18N')

chat), meaning that different people from different locations can collaborate at different times. Therefore, topics in an email thread may not change in sequential way. In the example, we see that topic 1 (i.e., 'telecon cancellation') is revisited after some gaps.

The headers (i.e., subjects) do not convey much information and are often misleading. In the example thread, participants use the same subject (i.e., 20030220 telecon) but they talk about 'responding to I18N' and 'TAG document' instead of 'telecon cancellation'. Writing style varies among participants, and many people tend to use informal, short and ungrammatical sentences. These properties of email limit the application of techniques that have been successful in monologues and dialogues.

LDA and *LCSeg* are the two state-of-the-art models for topic segmentation of multi-party conversation (e.g., (Galley et al., 2003), (Hsueh et al., 2006), (Georgescul et al., 2008)). In this section, at first we describe how the existing models of topic segmentation can be applied to emails. We then point out where these methods fail and propose extensions of these basic models for email conversations.

4.1 Latent Dirichlet Allocation (LDA)

Our first model is the probabilistic LDA model (Steyvers and Griffiths, 2007). This model relies on the fundamental idea that documents are mixtures of topics, and a topic is a multinomial distribution over words. The generative topic model specifies the following distribution over words within a document:

$$P(w_i) = \sum_{j=1}^{T} P(w_i|z_i = j)P(z_i = j)$$

Where T is the number of topics. $P(w_i|z_i = j)$ is the probability of word w_i under topic j and $P(z_i = j)$ is the probability that j^{th} topic was sampled for the i^{th} word token. We refer the multinomial distributions $\phi^{(j)} = P(w|z_i = j)$ and $\theta^{(d)} = P(z)$ as topic-word distribution and document-topic distribution respectively. (Blei et al., 2003) refined this basic model by placing a Dirichlet (α) prior on θ. (Griffiths and Steyvers, 2003) further refined it by placing a Dirichlet (β) prior on ϕ. The inference problem is to find ϕ and θ given a document set. Variational EM has been applied to estimate these two parameters directly. Instead of estimating ϕ and θ, one can also directly estimate the posterior distribution over $z = P(z_i = j|w_i)$ (topic assignments for words). One efficient estimation technique uses Gibbs sampling to estimate this distribution.

This framework can be directly applied to an email thread by considering each email as a document. Using LDA we get $z = P(z_i = j|w_i)$ (i.e., topic assignments for words). By assuming the words in a sentence occur independently we can estimate the topic assignments for sentences as follows:

$$P(z_i = j|s_k) = \prod_{w_i \in s_k} P(z_i = j|w_i)$$

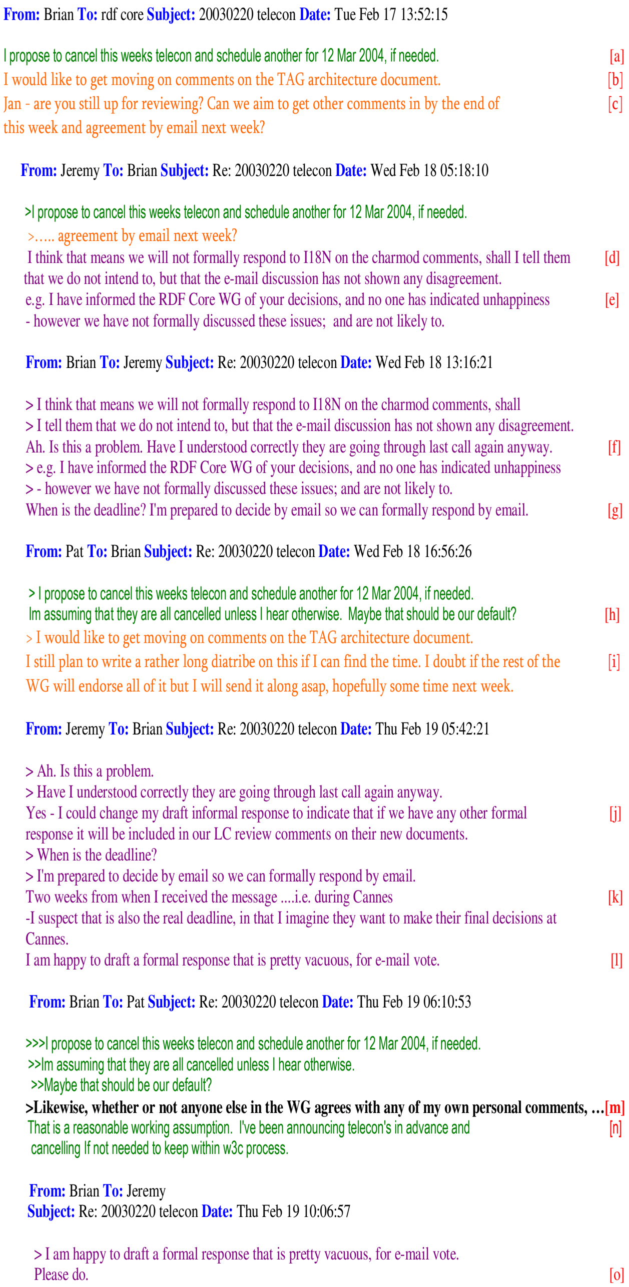
where, s_k is the k^{th} sentence for which we can assign the topic by: $j^* = argmax_j P(z_i = j|s_k)$.

4.2 Lexical Chain Segmenter (LCSeg)

Our second model is the lexical chain based segmenter LCSeg, (Galley et al., 2003). LCSeg assumes that topic shifts are likely to occur where strong term repetitions start and end[9]. LCSeg at first computes 'lexical chains' for each non-stop word based on word repetitions. It then ranks the chains according to two measures: 'number of words in the chain' and 'compactness of the chain'. The more compact (in terms of number of sentences) and the more populated chains get higher scores.

The algorithm then works with two adjacent analysis windows, each of a fixed size k which is empirically determined. For each sentence boundary, LCSeg computes the cosine similarity (or lexical cohesion function) at the transition between the two windows. Low similarity indicates low lexical cohesion, and a sharp change signals a high probability of an actual topic boundary. This method is similar to TextTiling (Hearst, 1997) except that the similarity is computed based on the scores of the 'lexical chains' instead of 'term counts'. In order to apply LCSeg on email threads we arrange the emails based on their temporal relation (i.e., arrival time) and apply the LCSeg algorithm to get the topic boundaries.

[9] One can also consider other lexical semantic relations (e.g., synonym, hypernym, hyponym) in lexical chaining. However, Galley et al., (Galley et al., 2003) uses only repetition relation as previous research results (e.g., (Choi, 2000)) account only for repetition.



From: Brian **To:** rdf core **Subject:** 20030220 telecon **Date:** Tue Feb 17 13:52:15

I propose to cancel this weeks telecon and schedule another for 12 Mar 2004, if needed. [a]
I would like to get moving on comments on the TAG architecture document. [b]
Jan - are you still up for reviewing? Can we aim to get other comments in by the end of this week and agreement by email next week? [c]

From: Jeremy **To:** Brian **Subject:** Re: 20030220 telecon **Date:** Wed Feb 18 05:18:10

>I propose to cancel this weeks telecon and schedule another for 12 Mar 2004, if needed.
>..... agreement by email next week?
I think that means we will not formally respond to I18N on the charmod comments, shall I tell them that we do not intend to, but that the e-mail discussion has not shown any disagreement. [d]
e.g. I have informed the RDF Core WG of your decisions, and no one has indicated unhappiness - however we have not formally discussed these issues; and are not likely to. [e]

From: Brian **To:** Jeremy **Subject:** Re: 20030220 telecon **Date:** Wed Feb 18 13:16:21

> I think that means we will not formally respond to I18N on the charmod comments, shall
> I tell them that we do not intend to, but that the e-mail discussion has not shown any disagreement.
Ah. Is this a problem. Have I understood correctly they are going through last call again anyway. [f]
> e.g. I have informed the RDF Core WG of your decisions, and no one has indicated unhappiness
> - however we have not formally discussed these issues; and are not likely to.
When is the deadline? I'm prepared to decide by email so we can formally respond by email. [g]

From: Pat **To:** Brian **Subject:** Re: 20030220 telecon **Date:** Wed Feb 18 16:56:26

> I propose to cancel this weeks telecon and schedule another for 12 Mar 2004, if needed.
Im assuming that they are all cancelled unless I hear otherwise. Maybe that should be our default? [h]
> I would like to get moving on comments on the TAG architecture document.
I still plan to write a rather long diatribe on this if I can find the time. I doubt if the rest of the WG will endorse all of it but I will send it along asap, hopefully some time next week. [i]

From: Jeremy **To:** Brian **Subject:** Re: 20030220 telecon **Date:** Thu Feb 19 05:42:21

> Ah. Is this a problem.
> Have I understood correctly they are going through last call again anyway.
Yes - I could change my draft informal response to indicate that if we have any other formal response it will be included in our LC review comments on their new documents. [j]
> When is the deadline?
> I'm prepared to decide by email so we can formally respond by email.
Two weeks from when I received the message ....i.e. during Cannes [k]
-I suspect that is also the real deadline, in that I imagine they want to make their final decisions at Cannes.
I am happy to draft a formal response that is pretty vacuous, for e-mail vote. [l]

From: Brian **To:** Pat **Subject:** Re: 20030220 telecon **Date:** Thu Feb 19 06:10:53

>>>I propose to cancel this weeks telecon and schedule another for 12 Mar 2004, if needed.
>>Im assuming that they are all cancelled unless I hear otherwise.
>>Maybe that should be our default?
>Likewise, whether or not anyone else in the WG agrees with any of my own personal comments, ...[m]
That is a reasonable working assumption. I've been announcing telecon's in advance and cancelling If not needed to keep within w3c process. [n]

From: Brian **To:** Jeremy
Subject: Re: 20030220 telecon **Date:** Thu Feb 19 10:06:57

> I am happy to draft a formal response that is pretty vacuous, for e-mail vote.
Please do. [o]

Figure 1: Sample thread from the BC3 corpus. Each different color/font indicates a different topic. Right most column specifies the fragments (sec 4.4).

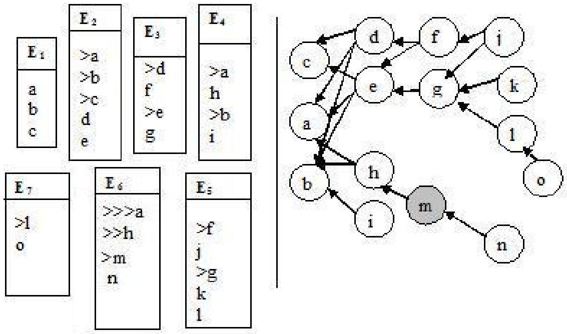


Figure 2: Fragment Quotation Graph for emails

4.3 Limitation of Existing Approaches

The main limitation of the two models discussed above is that they take the bag-of-words (BOW) assumption without considering the fact that an email thread is a multi-party, asynchronous conversation[10]. The only information relevant to LDA is term frequency. LCSeg considers both term frequency and how closely the terms occur in a document. These models do not consider the word order, syntax and semantics. However, several improvements of LDA over the BOW approach have been proposed. (Wallach, 2006) extends the model beyond BOW by considering n-gram sequences. (Griffiths et al., 2005) presents an extension of the topic model that is sensitive to word-order and automatically learns the syntactic as well as semantic factors that guide word choice. (Boyd-Graber and Blei, 2010) describes another extension to consider syntax of the text. As described earlier, one can also incorporate lexical semantics (i.e., synonym, hypernym, hyponym) into the LCSeg model. However, we argue that these models are still inadequate for finding topics in emails especially when topics are closely related (e.g., ‘extending the meeting’ and ‘scheduling the meeting’) and distributional variations are subtle. To better identify the topics in an email thread we need to consider the email specific conversation features (e.g., reply-to relation, usage of quotations). As can be seen in the example (Figure 1), people often use quotations to talk about the same topic. In fact in our corpus we found an average quotation usage of 6.44 per thread. Therefore,

[10] though in LCSeg we provide minimal conversation structure in the form of temporal relation between emails.

we need to leverage this useful information in a principled way to get the best out of our models. Specifically, we need to capture the conversation structure at the fragment (quotation) level and to incorporate this structure into our models.

In the next section, we describe how one can capture the conversation structure at the fragment level in the form of Fragment Quotation Graph (henceforth, FQG). In Section 4.5 and 4.6 respectively, we show how the LDA and LCSeg models can be extended so that they take this conversation structure into account for topic segmentation.

4.4 Extracting Conversation Structure

We demonstrate how to build a FQG through the example email thread involving 7 emails shown in Figure 1. For convenience we do not show the real content but abbreviate them as a sequence of fragments.

In the first pass by processing the whole thread we identify the new (i.e., quotation depth 0) and quoted (i.e., quotation depth $>$ 0) fragments based on the usage of quotation ($>$) marks. For instance, email E_3 contains two new fragments (f, g), and two quoted fragments (d, e) of depth 1. E_2 contains abc and de. Then in the second step, we compare the fragments with each other and based on the overlap we find the distinct fragments. If necessary we split the fragments in this step. For example, de in E_2 is divided into d and e distinct fragments when compared with the fragments of E_3. This process gives 15 distinct fragments which constitute the vertices of the FQG. In the third step, we compute the edges, which represent referential relations between fragments. For simplicity we assume that any new fragment is a potential reply to its neighboring quoted fragments. For example, for the fragments of E_4 we create two edges from h ((h,a),(h,b)) and one edge from i ((i,b)). We then remove the redundant edges. In E_6 we found the edges (n,h), (n,a) and (n,m). As (h,a) is already there we exclude (n,a). The FQG with all the redundant edges removed is shown at the right in Figure 2. If an email does not contain quotes then the fragments of that email are connected to the fragments of the source email to which it replies.

The advantage of the FQG is that it captures the conversation at finer granularity level in contrast to the structure found by the 'reply-to' relation at the email level, which would be merely a sequence from E_1 to E_7 in this example. Another advantage of this structure is that it allows us to find the 'hidden fragments'. Hidden fragments are quoted fragments (shaded fragment m in fig 2 which corresponds to the fragment made bold in fig 1), whose original email is missing in the user's inbox. (Carenini et al., 2007) study this phenomenon and its impact on email summarization in detail.

4.5 Regularizing LDA with FQG

The main advantage of the probabilistic (Bayesian) models is that they allow us to incorporate multiple knowledge sources in a coherent way in the form of priors (or regularizer). We want to regularize LDA in a way that will force two sentences in the same or adjacent fragments to fall in the same topical cluster. The first step forwards this aim is to regularize the topic-word distribution with a word network such that two connected words get similar topic distributions. Then we can easily extend it to fragments. In this section, at first we describe how one can regularize the LDA model with a word network, then we extend this by regularizing LDA with FQG.

Assume we are given a word network as an undirected graph with nodes (V) representing the words and the edges (E) representing the links between words. We want to regularize the LDA model such that two connected words u, v have similar topic-word distributions (i.e., $\phi_j^{(u)} \approx \phi_j^{(v)}$ for $j = 1 \dots T$). Note that the standard conjugate Dirichlet prior on ϕ is limited in that all words share a common variance parameter, and are mutually independent except normalization constraint (Minka, 1999). Therefore it does not allow us to encode this knowledge. Very recently, (Andrzejewski et al., 2009) shows how to encode 'must-link' and 'cannot-link' (between words) into the LDA model by using a Dirichlet Forest prior. We reimplemented this model; however, we only use its capability of encoding 'must-links'. Therefore, we just illustrate how to encode 'must-links' here. Interested readers can see (Andrzejewski et al., 2009) for the method of encoding 'cannot-links'.

Must links such as $(a, b), (b, c)$, or (x, y) in Figure 3(A) can be encoded into the LDA model by using a Dirichlet Tree (henceforth, DT) prior. Like the traditional Dirichlet, DT is also a conjugate to the multinomial but under a different parameterization.

Instead of representing a multinomial sample as the outcome of a K-sided die, in this representation we represent a sample as the outcome of a finite stochastic process. The probability of a leaf is the product of branch probabilities leading to that leaf. The words constitute the leaves of the tree.

DT distribution is the distribution over leaf probabilities. Let ω^n be the DT edge weight leading into node n, $C(n)$ be the children of node n, L be the leaves of the tree, I the internal nodes, and $L(n)$ be the leaves in the subtree under n. We generate a sample ϕ^k from Dirichlet_Tree(Ω) by drawing a multinomial at each internal node $i \in I$ from Dirichlet($\omega^{C(i)}$) (i.e., the edge weights from i to its children). The probability density function of DT($\phi^k|\Omega$) is given by:

$$DT(\phi^k|\Omega) \approx \left(\prod_{l\in L} \phi_l^{k^{\omega^l-1}}\right)\left(\prod_{i\in I}\left(\sum_{j\in L(i)} \phi_j^k\right)^{\Delta(i)}\right)$$

Here $\Delta(i) = \omega^i - \sum_{j\in C(i)} \omega^j$ (i.e., the difference between the in-degree and out-degree of internal node i. Note that if $\Delta(i) = 0$ for all $i \in I$, then the DT reduces to the typical Dirichlet distribution.

Suppose we have the following (Figure 3(A)) word network. The network can be decomposed into a collection of chains (e.g., (a,b,c), (p), and (x,y)). For each chain having number of elements more than one (e.g., (a,b,c), (x,y)), we have a subtree (see Figure 3(B)) in the DT with one internal node (blank in figure) and the words as leaves. We assign $\lambda\beta$ as the weights of these edges where λ is the regularization strength and β is the hyperparameter of the symmetric Dirichlet prior on ϕ. The root node of the Dirichlet tree then connects to the internal node i with weight $|L(i)|\beta$. The other nodes (words) which form single element chains (e.g, (p)) are connected to the root directly with weight β. Notice that when $\lambda = 1$ (i.e., no regularization), $\Delta(i) = 0$ and our model reduces to the original LDA. By tuning λ we control the strength of regularization.

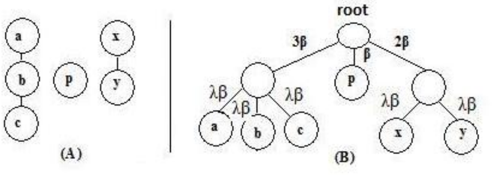


Figure 3: Incorporating word network into DT

To regularize LDA with FQG, we form the word network where a word is connected to the words in the same or adjacent fragments. Specifically, if word $w_i \in frag_x$ and word $w_j \in frag_y$ ($w_i \neq w_j$), we create a link (w_i, w_j) if $x = y$ or $(x, y) \in E$, where E is the set of edges of the FQG. Implicitly by doing this we want two sentences in the same or adjacent fragments to have similar topic distributions, and fall in the same topical cluster.

4.6 LCSeg with FQG

If we examine the FQG carefully, different paths (considering the fragments of the first email as root nodes) can be interpreted as subconversations. As we walk down a path topic shifts may occur along the pathway. We incorporate FQG into the LCSeg model in three steps. First, we extract the paths of a FQG. We then apply LCSeg algorithm on each of the extracted paths separately. This process gives the segmentation decisions along the paths of the FQG. Note that a fragment can be in multiple paths (e.g., f, g, in Figure 2) which will cause its sentences to be in multiple segments found by LCSeg. Therefore, as a final step we need a consolidation method. Our intuition is that sentences in a consolidated segment should fall in same segments more often when we apply LCSeg in step 2. To consolidate the segments found, we form a weighted undirected graph where the vertices V represent the sentences and the edge weights $w(u, v)$ represent the number of times sentence u and v fall in the same segment. The consolidation problem can be formulated as a *N-mincut* graph partitioning problem where we try to optimize the *Normalized Cut* criterion:

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(B, A)}{assoc(B, V)}$$

where $cut(A, B) = \Sigma_{u\in A, v\in B} w(u, v)$ and $assoc(A, V) = \Sigma_{u\in A, t\in V} w(u, t)$ is the total connection from nodes in partition A to all nodes in the graph and $assoc(B, V)$ is similarly defined. However, solving this problem turns out to be NP-hard. Hence, we approximate the solution following (Shi and Malik, 2000) which has been successfully applied to image segmentation in computer vision.

This approach makes a difference only if FGQ contains more than one path. In fact in our corpus we found an average paths of 7.12 per thread.

Avg. Topic	LDA	LDA +FQG	LCSeg	LCSeg +FQG	Speaker	Block 5
Number	2.10	1.90	2.2	2.41	4.87	5.69
Length	13.3	15.50	13.12	12.41	5.79	4.60
Density	1.83	1.60	1.01	1.39	1.37	1.00
Entropy	0.98	0.75	0.81	0.93	1.88	2.39

Table 3: Corpus statistics of different system's annotation

5 Experiments

We ran our four systems *LDA, LDA+FQG, LCSeg, and LCSeg+FQG* on the dataset[11]. The statistics of these four annotations and two best performing baselines (i.e., 'Speaker' and 'Block 5' as described below) are shown in Table 3. For brevity we just mention the average measures. Comparing with Table 1, we see that these fall within the bounds of the human annotations.

We compare our results in Table 4, where we also provide the results of some simple baseline systems. We evaluated the following baselines and report the best two in Table 4.

All different: Each sentence is a separate topic.

All same: The whole thread is a single topic.

Speaker: The sentences from each participant constitute a separate topic.

Blocks of $k (= 5, 10, 15)$**:** Each consecutive group of k sentences is a topic.

Most of these baselines perform rather poorly. **All different** is the worst baseline with mean 1-to-1 score of 0.10 (max: 0.33, min: 0.03) and mean loc_3 score of 0.245 (max: 0.67, min: 0). **Block 10** has mean 1-to-1 score of 0.35 (max: 0.71, min: 0.13) and mean loc_3 score of 0.584 (max: 0.76, min: 0.31). **Block 15** has mean 1-to-1 score of 0.32 (max: 0.77, min: 0.16) and mean loc_3 score of 0.56 (max: 0.82, min: 0.38). **All same** is optimal for threads containing only one topic, but its performance rapidly degrades as the number of topics in a thread increases. It has mean 1-to-1 score of 0.28 (max: 1[12], min: 0.11) and mean loc_3 score of 0.54 (max: 1, min: 0.34).

As shown in Table 4, **Speaker** and **Blocks of** 5 are two strong baselines especially for the loc_3. In general, our systems perform better than the baselines, but worse than the gold standard. Of all the systems, the basic LDA model performs very disappointingly. In the local agreement it even fails to beat the baselines. A likely explanation is that the independence assumption made by LDA when computing the distribution over topics for a sentence from the distribution over topics for the words causes sentences in a local context to be excessively distributed over topics. Another possible explanation for LDA's disappointing performance is the limited amount of data available for training. In our corpus, the average number of sentences per thread is 26.3 (see table 1) which might not be sufficient for the LDA models.

If we compare the performance of the regularized LDA (in the table LDA+FQG) with the basic LDA we get a significant (p=0.0002 (1-to-1), p=9.8e-07 (loc_3)) improvement in both of the measures [13]. This supports our claim that sentences connected by referential relations in the FQG usually refer to the same topic. The regularization also prevents the local context from being overly distributed over topics.

A comparison of the basic LCSeg with the basic LDA reveals that LCSeg is a better model for email topic segmentation (p=0.00017 (1-to-1), p<2.2e-16 (loc_3)). One possible reason is that LCSeg extracts the topics keeping the local context intact. Another reason could be the term weighting scheme employed by LCSeg. Unlike LDA, which considers only 'repetition', LCSeg also considers how tightly the 'repetition' happens. When we incorporate the conversation structure (i.e., FQG) into LCSeg (in the table LCSeg+FQG), we get a significant improvement in the 1-to-1 measure over the basic LCSeg (p=0.0014). Though the local context (i.e., loc_3) suf-

[11] For a fair comparison of the systems we set the same topic number per thread for all of them. If at least two of the annotators agree on the topic number we set that number, otherwise we set the floor value of the average topic number. λ is set to 20 in LDA+FQG.

[12] The maximum value of 1 is due to the fact that for some threads some annotators found only one topic

[13] Tests of significance were done by paired t-test with df=116

	Baselines		Systems				Human
Scores	Speaker	Block 5	LDA	**LDA+FQG**	LCSeg	**LCSeg+FQG**	
Mean 1-to-1	0.52	0.38	0.57	**0.62**	0.62	**0.68**	0.80
Max 1-to-1	0.94	0.77	1.00	1.00	1.00	1.00	1.00
Min 1-to-1	0.23	0.14	0.24	0.24	0.33	0.33	0.31
Mean loc_3	0.64	0.57	0.54	**0.61**	**0.72**	**0.71**	0.83
Max loc_3	0.97	0.73	1.00	1.00	1.00	1.00	1.00
Min loc_3	0.27	0.42	0.38	0.38	0.40	0.40	0.43

Table 4: Comparison of Human, System and best Baseline annotations

fers a bit, the decrease in performance is minimal and it is not significant. The fact that LCSeg is a better model than LDA is also preserved when we incorporate FQG into them (p=2.140e-05 (1-to-1), p=1.3e-09 (loc_3)). Overall, LCSeg+FQG is the best model for this data.

6 Future Work

There are some other important features that our models do not consider. The 'Speaker' feature is a key source of information. A participant usually contributes to the same topic. The best baseline 'Speaker' in Table 4 also favours this claim. Another possibly critical feature is the 'mention of names'. In multi-party discussion people usually mention each other's name for the purpose of disentanglement (Elsner and Charniak, 2008). In our corpus we found 175 instances where a participant mentions other participant's name. In addition to these, 'Subject of the email', 'topic-shift cue words' can also be beneficial for a model. As a next step for this research, we will investigate how to exploit these features in our methods.

We are also interested in the near future to transfer our approach to other similar domains by hierarchical Bayesian multi-task learning and other domain adaptation methods. We plan to work on both synchronous (e.g., chats, meetings) and asynchronous (e.g., blogs) domains.

7 Conclusion

In this paper we presented an email corpus annotated for topic segmentation. We extended LDA and LCSeg models by incorporating the fragment quotation graph, a fine-grain model of the conversation, which is based on the analysis of quotations. Empirical evaluation shows that the fragment quotation graph helps both these models to perform significantly better than their basic versions, with LCSeg+FQG being the best performer.

Acknowledgments

We are grateful to the 6 pilot annotators, 3 test annotators and to the 3 anonymous reviewers for their helpful comments. This work was supported in part by NSERC PGS award, NSERC BIN project, NSERC discovery grant and Institute for Computing, Information and Cognitive Systems (ICICS) at UBC.

References

James Allan, 2002. *Topic detection and tracking: event-based information organization*, pages 1–16. Kluwer Academic Publishers, Norwell, MA, USA.

David Andrzejewski, Xiaojin Zhu, and Mark Craven. 2009. Incorporating domain knowledge into topic modeling via dirichlet forest priors. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML'09)*, pages 25–32, New York, NY, USA. ACM.

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 sociable talk. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work (CSCW '06)*, pages 393–402, New York, NY, USA. ACM.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *JMLR*, 3:993–1022.

Jordan L. Boyd-Graber and David M. Blei. 2010. Syntactic topic models. *CoRR*, abs/1002.4665.

G. Carenini, R. T. Ng, and X. Zhou. 2007. Summarizing email conversations with clue words. In *Proceedings* *of the 16th international conference on World Wide Web*, pages 91–100. ACM New York, NY, USA.

Harr Chen, S. R. K. Branavan, Regina Barzilay, and David R. Karger. 2009. Global models of document structure using latent permutations. In *NAACL'09*, pages 371–379, Morristown, NJ, USA. ACL.

Freddy Y. Y. Choi, Peter Wiemer-Hastings, and Johanna Moore. 2001. Latent semantic analysis for text segmentation. In *In Proceedings of EMNLP*, pages 109–117, Pittsburgh, PA USA.

Freddy Y. Y. Choi. 2000. Advances in domain independent linear text segmentation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 26–33, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

David Crystal, 2001. *Language and the Internet*. Cambridge University Press.

Gaël Dias, Elsa Alves, and José Gabriel Pereira Lopes. 2007. Topic segmentation algorithms for text summarization and passage retrieval: an exhaustive evaluation. In *AAAI'07: Proceedings of the 22nd national conference on Artificial intelligence*, pages 1334–1339. AAAI Press.

Mark Dredze, Hanna M. Wallach, Danny Puller, and Fernando Pereira. 2008. Generating summary keywords for emails using topics. In *IUI '08*, pages 199–206, New York, NY, USA. ACM.

Micha Elsner and Eugene Charniak. 2008. You talking to me? a corpus and algorithm for conversation disentanglement. In *Proceedings of ACL-08: HLT*, pages 834–842, Ohio, June. ACL.

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.

M. Georgescul, A. Clark, and S. Armstrong. 2008. A comparative study of mixture models for automatic topic segmentation of multiparty dialogues. In *ACL-08:HLT*, pages 925–930, Ohio, June. ACL.

Thomas L. Griffiths and Mark Steyvers. 2003. Prediction and semantic association. In *Advances in Neural Information Processing Systems*. MIT Press.

Thomas L. Griffiths, Mark Steyvers, David M. Blei, and Joshua B. Tenenbaum. 2005. Integrating topics and syntax. In *In Advances in Neural Information Processing Systems*, pages 537–544. MIT Press.

Sanda Harabagiu and Finley Lacatusu. 2005. Topic themes for multi-document summarization. In *SIGIR '05:*, pages 202–209, New York, NY, USA. ACM.

Marti A. Hearst. 1997. Texttiling: segmenting text into multi-paragraph subtopic passages. *Comput. Linguist.*, 23(1):33–64, March.

Pei Hsueh, Johanna D. Moore, and Steve Renals. 2006. Automatic segmentation of multiparty dialogue. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*, Trento, Italy. ACL.

Igor Malioutov and Regina Barzilay. 2006. Minimum cut model for spoken lecture segmentation. In *Proceedings of the ACL'06*, pages 25–32, Sydney, Australia, July. ACL.

T. Minka. 1999. The dirichlet-tree distribution. Technical report, Justsystem Pittsburgh Research Center.

Matthew Purver, Konrad P. Körding, Thomas L. Griffiths, and Joshua B. Tenenbaum. 2006. Unsupervised topic modelling for multi-party spoken discourse. In *Proceedings of the ACL'06*, pages 17–24, Sydney, Australia. ACL.

Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905.

M. Steyvers and T. Griffiths, 2007. *Latent Semantic Analysis: A Road to Meaning*, chapter Probabilistic topic models. Laurence Erlbaum.

J. Ulrich, G. Murray, and G. Carenini. 2008. A publicly available annotated corpus for supervised email summarization. In *EMAIL-2008 Workshop*, pages 428–435. AAAI.

Hanna M. Wallach. 2006. Topic modeling: beyond bag-of-words. In *ICML '06*, pages 977–984, NY, USA.

A Semi-Supervised Approach to Improve Classification of Infrequent Discourse Relations using Feature Vector Extension

Hugo Hernault
hugo@mi.ci.i.u-tokyo.ac.jp

Danushka Bollegala
danushka@iba.t.u-tokyo.ac.jp

Mitsuru Ishizuka
ishizuka@i.u-tokyo.ac.jp

Graduate School of Information Science & Technology
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan

Abstract

Several recent discourse parsers have employed fully-supervised machine learning approaches. These methods require human annotators to beforehand create an extensive training corpus, which is a time-consuming and costly process. On the other hand, unlabeled data is abundant and cheap to collect. In this paper, we propose a novel semi-supervised method for discourse relation classification based on the analysis of co-occurring features in unlabeled data, which is then taken into account for extending the feature vectors given to a classifier. Our experimental results on the RST Discourse Treebank corpus and Penn Discourse Treebank indicate that the proposed method brings a significant improvement in classification accuracy and macro-average F-score when small training datasets are used. For instance, with training sets of c.a. 1000 labeled instances, the proposed method brings improvements in accuracy and macro-average F-score up to 50% compared to a baseline classifier. We believe that the proposed method is a first step towards detecting low-occurrence relations, which is useful for domains with a lack of annotated data.

1 Introduction

Automatic detection of discourse relations in natural language text is important for numerous tasks in NLP, such as sentiment analysis (Somasundaran et al., 2009), text summarization (Marcu, 2000) and dialogue generation (Piwek et al., 2007). However, most of the recent work employing discourse relation classifiers are based on fully-supervised machine learning approaches (duVerle and Prendinger, 2009; Pitler et al., 2009; Lin et al., 2009). Two of the main corpora with discourse annotations are the RST Discourse Treebank (RSTDT) (Carlson et al., 2001) and the Penn Discourse Treebank (PDTB) (Prasad et al., 2008a), which are both based on the Wall Street Journal (WSJ) corpus.

In the RSTDT, annotation is done using 78 fine-grained discourse relations, which are usually grouped into 18 coarser-grained relations. Each of these relations has furthermore several possible configurations for its arguments—its 'nuclearity' (Mann and Thompson, 1988). In practice, a classifier trained on these coarse-grained relations must solve a 41-class classification problem. Some of the relations corresponding to these classes are relatively more frequent in the corpus, such as the ELABORATION[N][S] relation (4441 instances), or the ATTRIBUTION[S][N] relation (1612 instances).[1] However, other relation types occur very rarely, such as TOPIC-COMMENT[S][N] (2 instances), or EVALUATION[N][N] (3 instances). A similar phenomenon can be observed in PDTB, in which 15 level-two relations are employed: Some, such as EXPANSION.CONJUNCTION, occur as often as 8759 times throughout the corpus, whereas the remainder of the relations, such as EXPANSION.EXCEPTION and COMPARISON.PRAGMATIC CONCESSION, can appear as rarely as 17 and 12 times respectively. Although supervised approaches to discourse relation learning achieve good results on frequent relations, performance is poor on rare relation types (duVerle and Prendinger, 2009).

Nonetheless, certain infrequent relation types might be important for specific tasks. For instance,

[1] We use the notation [N] and [S] respectively to denote the nucleus and satellite in a RST discourse relation.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 399–409,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

capturing the RST TOPIC-COMMENT[S][N] and EVALUATION[N][N] relations can be useful for sentiment analysis (Pang and Lee, 2008).

Another situation where detection of low-occurring relations is desirable is the case where we have only a small training set at our disposal, for instance when there is not enough annotated data for all the relation types described in a discourse theory. In this case, all the dataset's relations can be considered rare, and being able to build an efficient classifier depends on the capacity to deal with this lack of annotated data.

Our contributions in this paper are summarized as follows.

- We propose a semi-supervised method that exploits the abundant, freely-available unlabeled data, which is harvested for feature co-occurrence information, and used as a basis to extend feature vectors to help classification for cases where unknown features are found in test vectors.

- The proposed method is evaluated on the RSTDT and PDTB corpus, where it significantly improves accuracy and macro-average F-score when small training sets are used. For instance, when trained on moderately small datasets with ca. 1000 instances, the proposed method increases the macro-average F-score and accuracy up to 50%, compared to a baseline classifier.

2 Related Work

Since the release in 2001 of the RSTDT corpus, several fully-supervised discourse parsers have been built in the RST framework. In the recent work of duVerle and Prendinger (2009), a discourse parser based on Support Vector Machines (SVM) (Vapnik, 1995) is proposed. SVMs are employed to train two classifiers: One, binary, for determining the presence of a relation, and another, multi-class, for determining the relation label between related text spans. For the discourse relation classifier, shallow lexical, syntactic and structural features, including 'dominance sets' (Soricut and Marcu, 2003) are used. For relation classification, they report an accuracy of 0.668, and an F-score of 0.509 for the creation of the full discourse tree.

The unsupervised method of Marcu and Echihabi (2002) was the first that tried to detect implicit relations (i.e. relations not accompanied by a cue phrase, such as *'however'*, *'but'*), using word pairs extracted from two spans of text. Their method attempts to capture the difference of polarity in words. For example, the word pair *(sell, hold)* indicates a CONTRAST relation.

Discourse relation classifiers have also been trained using PDTB. Pitler et al. (2008) performed a corpus study of the PDTB, and found that 'explicit' relations can be most of the times distinguished by their discourse connectives. Their discourse relation classifier reported an accuracy of 0.93 for explicit relations and in overall an accuracy of 0.744 for all relations in PDTB.

Lin et al. (2009) studied the problem of detecting implicit relations in PDTB. Their relational classifier is trained using features extracted from dependency paths, contextual information, word pairs and production rules in parse trees. They reported for their classifier an accuracy of 0.402, which is an improvement of 14.1% over the previous state-of-the-art for implicit relation classification in PDTB. For the same task, Pitler et al. (2009) also used word pairs, as well as several other types of features such as verb classes, modality, context, and lexical features.

In text classification, similarity measures have been employed in kernel methods, where they have been shown to improve accuracy over 'bag-of-words' approaches. In Siolas and d'Alché-Buc (2000), a semantic proximity measure based on WordNet (Fellbaum, 1998) is defined, as a basis to create a proximity matrix for all terms of the problem. This matrix is then used to smooth the vectorial data, and the resulting 'semantic' metric is incorporated into a SVM kernel, resulting in a significant increase of accuracy and F-score over a baseline.

Cristianini et al. (2002) have used a lexical similarity measure derived from Latent Semantic Indexing (Deerwester et al., 1990), where the semantic similarity between two terms is inferred from the analysis of their co-occurrence patterns: Terms that co-occur often in the same documents are considered as related. In this work, the statistical co-occurrence information is extracted by the means of singular value decomposition. The authors observe substantial improvements in performance for some datasets, while little effect is obtained for others.

Semantic kernels have also been shown to be efficient for text classification tasks, in the case in of unbalanced and sparse datasets. In Basili et al. (2006), a ‘conceptual density’ metric based on WordNet is introduced, and employed in a SVM kernel. Using this metric results in improved accuracy of 10% for text classification in poor training conditions. However, the authors observe that when the number of training documents is increased, the improvement produced by the semantic kernel is lower.

Bloehdorn et al. (2006) compare the performance of different semantic kernels, based on several measures of semantic relatedness in WordNet. For each measure, the authors note a performance increase when little training data is available, or when the feature representations are very sparse. However, for our task, classification of discourse relations, we employ not only words but also other types of features such as parse tree production rules, and thus cannot compute semantic kernels using WordNet.

In this paper, we are not aiming at defining novel features for improving performance in RST or PDTB relation classification. Instead we incorporate numerous features that have been shown to be useful for discourse relation learning and explore the possibilities of using unlabeled data for this task. One of our goals is to improve classification accuracy for rare discourse relations.

3 Method

Given a set of unlabeled instances U and labeled instances L, our objective is to learn an n-class relation classifier H such that for a given test instance $\mathbf{x}$ return its correct relation type $H(\mathbf{x})$. In the case of discourse relation learning we are interested in the situation where $|U| >> |L|$. Here, we use the notation $|A|$ to denote the number of elements in a set A. A fundamental problem that one encounters when trying to learn a classifier for a large number of relations with small training dataset is that most of the features that appear in the test instances either never occur in training instances or appear a small number of times. Therefore, the classification algorithm does not have sufficient information to correctly predict the relation type of the given test instance. We propose a method that first computes the co-occurrence between features using unlabeled data and use that information to *extend* the feature vectors during training and testing, thereby reducing the sparseness in test feature vectors. In Section 3.1, we introduce the concept of *feature co-occurrence matrix* and describe how it is computed using unlabeled data. A method to extend feature vectors during training and testing is presented in Section 3.2. We defer the details on exact features used in the method to Section 3.3. It is noteworthy that the proposed method does not depend or assume a particular multi-class classification algorithm. Consequently, it can be used with any multi-class classification algorithm to learn a discourse relation classifier.

3.1 Feature Co-occurrence Matrix

We represent an instance using a d dimensional feature vector $\mathbf{f} = [f_1, \ldots, f_d]^{\mathrm{T}}$, where $f_i \in \mathbb{R}$. We define a *feature co-occurrence matrix*, C such that the (i, j)-th element of C, $C_{(i,j)} \in [0, 1]$ denotes the degree of co-occurrence between the two features f_i and f_j. If both f_i and f_j appear in a feature vector then we define them to be co-occurring. The number of different feature vectors in which f_i and f_j co-occur is denoted by the function $h(f_i, f_j)$. From our definition of co-occurrence it follows that $h(f_i, f_j) = h(f_j, f_i)$. Importantly, feature co-occurrences can be calculated only using unlabeled data.

Feature co-occurrence matrices can be computed using any co-occurrence measure. For the current task we use the χ^2-measure (Plackett, 1983) as the preferred co-occurrence measure because of its simplicity. χ^2-measure between two features f_i and f_j is defined as follows,

$$\chi^2_{i,j} = \sum_{k=1}^{2}\sum_{l=1}^{2} \frac{(O^{i,j}_{k,l} - E^{i,j}_{k,l})^2}{E^{i,j}_{k,l}}. \qquad (1)$$

Therein, $O^{i,j}$ and $E^{i,j}$ are the 2×2 matrices containing respectively observed frequencies and expected frequencies, which are respectively computed using C as,

$$O^{i,j} = \begin{pmatrix} h(f_i, f_j) & Z_i - h(f_i, f_j) \\ Z_j - h(f_i, f_j) & Z_s - Z_i - Z_j \end{pmatrix}, \qquad (2)$$

and

$$E^{i,j} = \begin{pmatrix} \frac{Z_i \cdot Z_j}{Z_s} & \frac{Z_i \cdot (Z_s - Z_j)}{Z_s} \\ \frac{Z_j \cdot (Z_s - Z_i)}{Z_s} & \frac{(Z_s - Z_i) \cdot (Z_s - Z_j)}{Z_s} \end{pmatrix}. \quad (3)$$

Here, $Z_i = \sum_{k \neq i} h(f_i, f_k)$, and $Z_s = \sum_{i=1}^{n} Z_i$.

Finally, we create the feature co-occurrence matrix C, such that, for all pairs of features (f_i, f_j),

$$C_{(i,j)} = \begin{cases} \hat{\chi}^2_{i,j} \text{ if } \chi^2_{i,j} > c \\ 0 \text{ otherwise} \end{cases}. \quad (4)$$

Here $\hat{\chi}^2_{i,j} = \frac{\chi^2_{i,j} - \chi^2_{min}}{\chi^2_{max} - \chi^2_{min}} \in [0, 1]$, and c is the critical value, which, for a confidence level of 0.05 and one degree of freedom, can be set to 3.84. Keeping $C_{(i,j)}$ in the range $[0, 1]$ makes it convenient to filter out low-relevance co-occurrences at the feature vector extension step of Section 3.2.

In discourse relation learning, the feature space can be extremely large. For example, with word pair features (discussed later in Section 3.3), any two words that appear in two adjoining discourse units can form a feature. Because the number of elements in the feature co-occurrence matrix is proportional to the square of the feature space's dimension, computing co-occurrences for all pairs of features can be computationally costly. Moreover, storing a large matrix in memory for further computations can be problematic. To reduce the dimensionality and improve the sparseness in the feature co-occurrence matrix, we use entropy-based feature selection (Manning and Schütze, 1999). The negative entropy, $E(f_i)$, of a feature f_i is defined as follows,

$$\mathrm{E}(f_i) = -\sum_{j \neq i} p(i,j) \cdot \log\left(p\left(i,j\right)\right). \quad (5)$$

Here, $p(i,j)$ is the probability that feature f_i co-occurs with feature f_j, and is given by $p(i,j) = h(f_i, f_j)/Z_i$.

If a particular feature f_i co-occurs with many other features, then its negative entropy $E(f_i)$ decreases. Because we are interested in identifying salient co-occurrences between features, we can ignore the features that tend to co-occur with many other features. Consequently, we sort the features in the descending order of their entropy, and select the top ranked N number of features to build the feature co-occurrence matrix. This feature selection procedure can efficiently reduce the dimensions of the feature co-occurrence matrix to $N \times N$. Because the feature co-occurrence matrix is symmetric, we must only store the elements for the upper (or lower) triangular portion of it.

3.2 Feature Vector Extension

Once the feature co-occurrence matrix is computed using unlabeled data as described in Section 3.1, we can use it to extend a feature vector during training and testing. The proposed feature vector extension method is inspired by query expansion in the field of Information Retrieval (Salton and Buckley, 1983; Fang, 2008). One of the reasons that a classifier might perform poorly on a test instance is that there are features in the test instance that were not observed during training. We call $F_U = \{f_i\}$ the set of features that were not observed by the classifier during training (i.e. occurring in test data but not in training data). For each of those features, we use the feature co-occurrence matrix to find the set of co-occurring features, $F_c(f_i)$.

Let us denote the feature vector corresponding to a training or test instance x by $\mathbf{f}_x$. We use the superscript notation, f_x^i to denote the i-th feature in $\mathbf{f}_x$. Moreover, the total number of features of $\mathbf{f}_x$ is indicated by $d(x)$. For a feature f_x^i in $\mathbf{f}_x$, we define $n(i)$ number of *expansion features*, $f_x^{(i,1)}, \ldots, f_x^{(i,n(i))}$ as follows. First, we require that each expansion feature $f_x^{(i,j)}$ belongs to $F_c(f_i)$. Second, the value of $f_x^{(i,j)}$ is set to $f_x^i \cdot C_{(i,j)}$. The expansion features for each feature f_x^i are then appended to the original feature vector $\mathbf{f}_x$ to create an extended feature vector, $\mathbf{f}'_x$, where,

$$\begin{aligned} \mathbf{f}'_x \quad = \quad & (f_x^1, \ldots, f_x^{d(x)}, \\ & f_x^{(i,1)}, \ldots, f_x^{(i,n(i))}, \ldots, \\ & f_x^{(d(x),1)}, \ldots, f_x^{(d(x),n(d(x)))}). \end{aligned} \quad (6)$$

In total, doing so augments the original vector's size by $\sum_{f_i \in U} |F_c(f_i)|$. All training and test instances are extended in this fashion.

Note that because this process can potentially increase the dimension too much, it is possible to retain only candidate co-occurring features of $F_c(f_i)$ possessing a co-occurrence value $C_{(i,j)}$ above a certain threshold. In the experiments of Section 4 however, we experienced dimension increase of 10000 at most, which did not require us to use thresholding.

3.3 Features

We use three types of features: Word pairs, production rules from the parse tree, as well as features encoding the lexico-syntactic context at the border between two units of text (Soricut and Marcu, 2003). Our word pairs are lemmatized using the Wordnet-based lemmatizer of NLTK (Loper and Bird, 2002).

Figure 1 shows the parse tree for a sentence composed of two discourse units, which serve as arguments of a discourse relation we want to generate a feature vector from. Lexical heads have been calculated using the projection rules of Magerman (1995), and annotated between brackets. Surrounded by dots is, for each argument, the minimal set of sub-parse trees containing strictly all the words of the argument.

We first extract all possible lemmatized word-pairs from the two arguments, such as *(Mr., when)*, *(decline, ask)* or *(comment, sale)*. Next, we extract from left and right argument separately, all production rules from the sub-parse trees, such as NP $\mapsto$ NNP NNP, NNP $\mapsto$ "Sherry" or TO $\mapsto$ "to".

Finally, we encode in our features three nodes of the parse tree, which capture the local context at the connection point between the two arguments: The first node, which we call N_w, is the highest ancestor of the first argument's last word w, and is such that N_w's right-sibling is the ancestor of the second argument's first word. N_w's right-sibling node is called N_r. Finally, we call N_p the parent of N_w and N_r. For each node, we encode in the feature vector its part-of-speech (POS) and lexical head. For instance, in Figure 1, we have N_w = S(comment), N_r = SBAR(when), and N_p = VP(declined). In the PDTB, certain discourse relations have disjoint arguments. In this case, as well as in the case where the two arguments belong to different sentences, the nodes N_w, N_r, N_p cannot be defined, and their corresponding features are given the value zero.

4 Experiments

The proposed method is independent of any particular classification algorithm. Because our goal is strictly to evaluate the relative benefit of employing the proposed method, and not the absolute performance when used with a specific classification algorithm, we select a logistic regression classifier, for its simplicity. We use the multi-class logistic regression (maximum entropy model) implemented in the Classias toolkit (Okazaki, 2009). Regularization parameters are set to their default value of one and are fixed throughout the experiments described in the paper.

To create our unlabeled dataset, we use sentences extracted from the English Wikipedia[2], as they are freely available and relatively easy to collect. For further extraction of syntactic features, these sentences are automatically parsed using the Stanford parser (Klein and Manning, 2003). Then, they are segmented into elementary discourse units (EDUs) using our sequential discourse segmenter (Hernault et al., 2010). The relatively high performance of this RST segmenter, which has an F-score of 0.95 compared to that of 0.98 between human annotators (Soricut and Marcu, 2003), is acceptable for this task. We collect and parse 100000 sentences from random Wikipedia articles. As there is no segmentation tool for the PDTB framework, we assume that co-occurrence information taken from EDUs created using a RST segmenter is also useful for extending feature vectors of PDTB relations. Unless otherwise noted, the experiments presented in the rest of this paper are done using those 100000 unlabeled instances.

In the unlabeled data, any two consecutive discourse units might not always be connected by a discourse relation. Therefore, we introduce an artificial NONE relation in the training set, in order to facilitate this. Instances of the NONE relation are generated randomly by pairing consecutive discourse units which are not connected by a discourse relation in the training data. NONE is also learnt as a separate discourse relation class by the multi-class classification algorithm. This enables us to detect discourse units between which there exist no discourse relation, thereby improving the classification accuracy for other relation types.

We follow the common practice in discourse research for partitioning the discourse corpora into training and test set. For the RST classifier, the dedicated training and test sets of the RSTDT are

[2] `http://en.wikipedia.org`

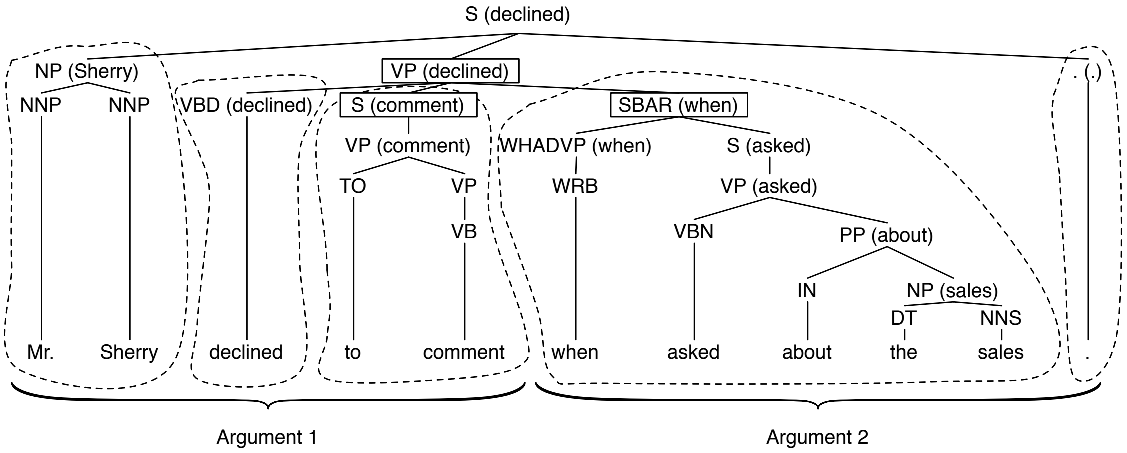


Figure 1: Two arguments of a discourse relation, and the minimum set of subtrees that contain them—lexical heads are indicated between brackets.

employed. For the PDTB classifier, we conform to the guidelines of Prasad et al. (2008b, 5): The portion of the corpus corresponding to sections 2–21 of the WSJ is used for training the classifier, while the portion corresponding to WSJ section 23 is used for testing. In order to extract syntactic features, all training and test data are furthermore aligned with their corresponding parse trees in the Penn Treebank (Marcus et al., 1993).

Because in the PDTB an instance can be annotated with several discourse relations simultaneously—called 'senses' in Prasad et al. (2008b)—for each instance with n senses in the corpus, we create n identical feature vectors, each being labeled by one of the instance's senses. However, in the RST framework, only one relation is allowed to hold between two EDUs. Consequently, each instance from the RSTDT is labeled with a single discourse relation, from which a single feature vector is created. For RSTDT, we extract 25078 training vectors and 1633 test vectors. For PDTB we extract 49748 training vectors and 1688 test vectors. There are 41 classes (relation types) in the RSTDT relation classification task, and 29 classes in the PDTB task. For the PDTB, we selected level-two relations, because they have better expressivity and are not too fine-grained. We experimentally set the entropy-based feature selection parameter to $N = 5000$. With large N values, we must store and process large feature co-occurrence matrices. For example, doubling the number of selected features, N to 10000 did not improve the classification accuracy, although it required 4GB of memory to store the feature co-occurrence matrix.

Figure 2 shows the number of features that occur in test data but not in labeled training data, against the number of training instances. It can be seen from Figure 2 that, with less training data available to the classifier, we can potentially obtain more information regarding features by looking at unlabeled data. However, when the training dataset's size increases, the number of features that only appear in test data decreases rapidly. This inverse relation between the training dataset size and the number of features that only appear in test data can be observed in both RSTDT and PDTB datasets. For a training set of 100 instances, there are 23580 unseen features in the case of RSTDT, and 27757 in the case of PDTB. The number of unseen features is halved for a training set of 1800 instances in the case of RSTDT, and for a training set of 1300 instances in the case of PDTB. Finally, when selecting all available training data, we count only 1365 unseen test features in the case of RSTDT, and 87 in the case of PDTB.

In the following experiments, we use macro-averaged F-scores to evaluate the performance of the proposed discourse relation classifier on test data. Macro-averaged F-score is not influenced by the number of instances that exist in each relation type. It equally weights the performance on both frequent relation types and infrequent relation types. Because we are interested in measuring the overall performance of a discourse relation classifier across all re-

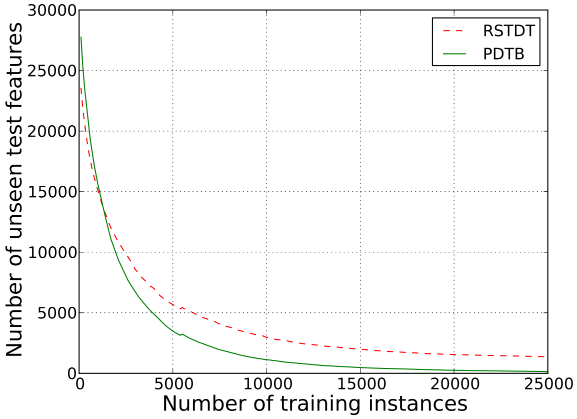


Figure 2: Number of features seen only in the test set, as a function of the number of training instances used.

lation types we use macro-averaged F-score as the preferred evaluation metric for this task.

We train a multi-class logistic regression model without extending the feature vectors as a baseline method. This baseline is expected to show the effect of using the proposed feature vector extension approach for the task of discourse relation learning. Experimental results on RSTDT and PDTB datasets are depicted in Figures 3 and 4. From these figures, we see that the proposed feature extension method outperforms the baseline for both RSTDT and PDTB datasets for the full range of training dataset sizes. However, whereas the difference of scores between the two methods is obvious for small amounts of training data, this difference progressively decreases as we increase the amount of training data. Specifically, with 100 training instances, the difference between baseline and proposed method is the largest: For RSTDT, the baseline has a macro-averaged F-score of 0.084, whereas the the proposed method has a macro-averaged F-score of 0.189 (ca. 119% increase in F-score). For PDTB, the baseline has an F-score of 0.016, while the proposed method has an F-score of 0.089 (459% increase). The difference of scores between the two methods then progressively diminishes as the number of training instances is increased, and fades beyond 10000 training instances. The reason for this behavior is given by Figure 2: For a small number of training instances, the number of unseen features in training data is large. In this case, the feature vector extension process is comprehensive, and score can be increased by the use of unlabeled data. When more training data is progressively used, the number of unseen test features sharply diminishes, which means feature vector extension becomes more limited, and the performance of the proposed method gets progressively closer to the baseline. Note that we plotted PDTB performance up to 25000 training instances, as the number of unseen test features becomes so small past this point that the performances of the proposed method and baseline are identical. Using all PDTB training data (49748 instances), both baseline and proposed method reach a macro-average F-score of 0.308.

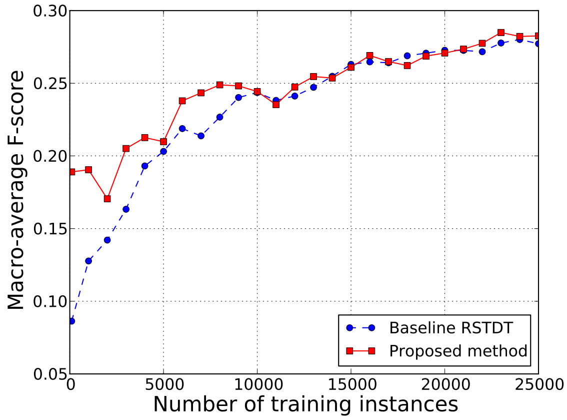


Figure 3: Macro-average F-score (RSTDT) as a function of the number of training instances used.

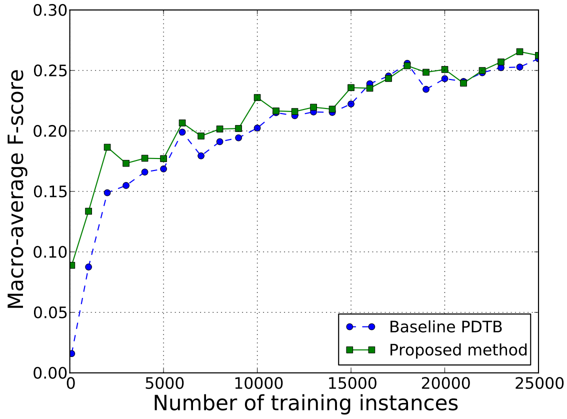


Figure 4: Macro-average F-score (PDTB) as a function of the number of training instances used.

	$\#$**Tr** $=$ **1**		$\#$**Tr** $=$ **2**		$\#$**Tr** $=$ **3**		$\#$**Tr** $=$ **5**		$\#$**Tr** $=$ **7**	
Relation name	**B.**	**P.M.**	**B.**	**P.M.**	**B.**	**P.M.**	**B.**	**P.M.**	**B.**	**P.M.**
Attribution[N][S]	–	**0.127**	–	**0.237**	–	**0.458**	0.038	**0.290**	0.724	**0.773**
Attribution[S][N]	–	**0.597**	–	**0.449**	0.009	**0.639**	0.250	**0.721**	0.579	**0.623**
Background[N][S]	–	**0.113**	–	**–**	–	**0.036**	–	**0.095**	–	**0.089**
Cause[N][S]	–	**–**	–	**0.128**	–	**–**	–	**0.034**	0.057	**0.187**
Comparison[N][S]	–	**0.118**	–	**0.037**	–	**–**	**0.133**	0.130	**0.143**	0.031
Condition[N][S]	–	**0.041**	–	**0.136**	–	**0.113**	–	**0.154**	**0.242**	0.152
Condition[S][N]	–	**–**	–	**0.122**	0.133	**0.148**	0.214	**0.233**	**0.390**	0.308
Contrast[N][N]	–	**–**	–	**0.086**	–	**0.073**	0.050	**0.111**	–	**0.109**
Contrast[N][S]	–	**0.071**	–	**–**	–	**0.188**	–	**0.087**	–	**0.136**
Elaboration[N][S]	–	**0.134**	–	**0.126**	0.004	**0.067**	0.004	**0.340**	–	**0.165**
Enablement[N][S]	–	**–**	–	**0.462**	–	**0.579**	0.115	**0.423**	0.419	**0.438**
Joint[N][N]	–	**0.030**	–	**0.015**	–	**–**	0.016	**0.059**	0.015	**0.155**
Manner-Means[N][S]	–	**–**	–	**0.056**	–	**0.103**	0.345	**0.372**	**0.412**	0.383
Summary[N][S]	–	**0.429**	–	**0.453**	0.080	**0.358**	–	**0.349**	0.154	**0.471**
Temporal[N][S]	–	**0.158**	–	**–**	–	**0.091**	–	**0.052**	**0.204**	0.101
Accuracy	0.000	**0.110**	0.000	**0.105**	0.004	**0.146**	0.034	**0.222**	0.122	**0.213**
Macro-average F-score	0.000	**0.060**	0.000	**0.069**	0.008	**0.101**	0.038	**0.118**	0.107	**0.134**

Table 1: F-scores for RSTDT relations, using a training set containing #Tr instances of each relation. B. indicates F-score for baseline, P.M. for the proposed method. A boldface indicates the best classifier for each relation.

Although the distribution of discourse relations in RSTDT and PDTB is not uniform, it is possible to study the performance of the proposed method when all relations are made equally rare. We evaluate performance on artificially-created training sets containing an equal amount of each discourse relation. Table 1 contains the F-score for each RSTDT relation, using training sets containing respectively one, two, three, five and seven instances of each relation. For space considerations, only relations with significant results are shown. We observe that, when using respectively one and two instances of each relation, the baseline classifier is unable to detect any relation, and has a macro-average F-score of zero. Contrastingly, the classifier built with feature vector extension reaches in those cases an F-score of 0.06. Furthermore, when employing the proposed method, certain relations have relatively high F-scores even with very little labeled data: With one training instance, ATTRIBUTION[S][N] has an F-score of 0.597, while SUMMARY[N][S] has an F-score of 0.429. With three training instances, ENABLEMENT[N][S] has an F-score of 0.579. When the amount of each relation is increased, the baseline classifier starts detecting more relations. In all cases, the proposed method performs better in terms of accuracy and macro-average F-score. With a training set containing seven instances of each relation, the baseline's macro-average F-score is starting to get closer to the extended classifier's, with superior performances for several relations, such as COMPARISON[N][S], CONDITION[N][S], and TEMPORAL[N][S]. Still, in this case, the extended classifier's accuracy is higher than the baseline (0.213 versus 0.122). Table 2 summarizes the outcome of the same experiments performed on the PDTB dataset. The results exhibit a similar trend, despite the baseline classifier having a relatively high accuracy for each case.

Using the data from Figures 2, 3 and 4, it is possible to calculate the relative score change occurring when using the proposed method, as a function of the number of unseen features found in test data. This graph is plotted in Figure 5. Besides macro-average F-score, we additionally plot accuracy change. In the top subfigure, representing the case of RSTDT, we see that, for the lowest amount of unseen test features, the proposed method does

	$\#\mathbf{Tr} = \mathbf{1}$		$\#\mathbf{Tr} = \mathbf{2}$		$\#\mathbf{Tr} = \mathbf{3}$		$\#\mathbf{Tr} = \mathbf{5}$		$\#\mathbf{Tr} = \mathbf{7}$	
Relation name	**B.**	**P.M.**	**B.**	**P.M.**	**B.**	**P.M.**	**B.**	**P.M.**	**B.**	**P.M.**
Comparison.Concession[2][1]	–	**0.056**	–	–	–	**0.133**	–	–	–	**0.154**
Comparison.Contrast[2][1]	–	–	–	**0.333**	–	–	–	**0.190**	0.105	**0.368**
Contingency.Cause[1][2]	–	**0.013**	–	**0.007**	–	–	–	**0.026**	–	**0.013**
Contingency.Condition[1][2]	–	**0.082**	–	**0.160**	–	**0.127**	0.250	**0.253**	**0.214**	0.171
Contingency.Condition[2][1]	–	–	–	–	–	**0.074**	–	**0.143**	0.250	**0.296**
Contingency.Prag. cond.[1][2]	–	–	–	**0.133**	–	**0.034**	–	–	**0.133**	0.043
Contingency.Prag. cond.[2][1]	–	–	–	–	–	–	**0.133**	0.087	**0.154**	0.087
Expansion.Conjunction[1][2]	0.326	**0.352**	0.326	**0.351**	0.326	**0.368**	0.332	**0.371**	0.335	**0.384**
Expansion.Instantiation[1][2]	–	–	–	–	–	**0.042**	–	**0.057**	–	**0.131**
Temporal.Asynchronous[1][2]	–	**0.204**	–	–	–	**0.142**	0.039	**0.148**	–	**0.035**
Temporal.Asynchronous[2][1]	–	–	–	–	–	**0.316**	–	**0.483**	**0.143**	–
Temporal.Synchrony[1][2]	–	–	–	**0.032**	–	**0.162**	0.032	**0.103**	0.032	**0.157**
Temporal.Synchrony[2][1]	–	–	–	**0.083**	–	**0.143**	0.200	**0.308**	**0.211**	0.174
Accuracy	0.195	**0.201**	0.195	**0.202**	0.195	**0.212**	0.202	**0.214**	0.204	**0.213**
Macro-average F-score	0.015	**0.033**	0.015	**0.054**	0.015	**0.084**	0.045	**0.108**	0.072	**0.100**

Table 2: F-scores for PDTB relations.

not bring any change in F-score or accuracy. Indeed, as the number of unknown features is low, feature vector extension is very limited, and does not improve the performance compared to the baseline. Then, a progressive increase of both accuracy and macro-average F-score is observed, as the number of unseen test features is incremented. For instance, for 8500 unseen test features, the macro-average F-score increase (resp. accuracy increase) is 25% (resp. 2.5%), while it is 20% (resp. 1%) for 11000 unseen test instances. These values reach a maximum of 119% macro-average F-score increase, and 66% accuracy increase, when 23500 features unseen during training are present in test data. This situation corresponds in Figures 3 and 4 to the case of very small training sets. The bottom subfigure of Figure 2, for the case of PDTB, reveals a similar tendency. The macro-average F-score increase (resp. accuracy increase) is negligible for 1000 unseen test features, while this increase is 21% for both macro-average F-score and accuracy in the case of 9700 unseen test features, and 459% (resp. 630% for accuracy) when 28000 unseen features are found in test data. This shows that the proposed method is useful when large numbers of features are missing from the training set, which corresponds in practice to small training sets, with few training instances for each relation type. For large training sets, most features are encountered by the classifier during training, and feature vector extension does not bring useful information.

We empirically evaluate the effect of using different amounts of unlabeled data on the performance of the proposed method. We use respectively 100 and 10000 labeled training instances, create feature co-occurrence matrices with different amounts of unlabeled data, and evaluate the performance in relation classification. Experimental results for RSTDT are illustrated in Figure 6 (top). From Figure 6 it appears clearly that macro-average F-scores improve with increased number of unlabeled instances. However, the benefit of using larger amounts of unlabeled data is more pronounced when only a small number of labeled training instances are employed (ca. 100). In fact, with 100 labeled training instances, the maximum improvement in F-score is 119% (corresponds to using all our 100000 unlabeled instances). However, the maximum improvement in F-score with 10000 labeled training instances is small, only 2.5% (corresponds to 10000 unlabeled instances).

The effect of using unlabeled data on PDTB relation classification is illustrated in Figure 6 (bottom). Similarly, we consecutively set the labeled training dataset size to 100 and 10000 instances, and plot the macro-average F-score against the unlabeled dataset size. As in the RSTDT experiment, the benefit of us-

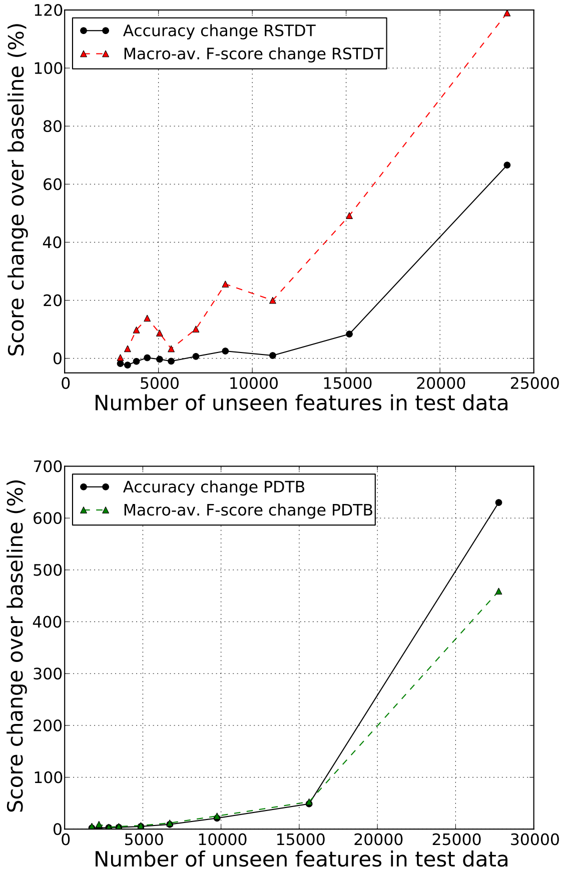


Figure 5: Score change as a function of unseen test features for RSTDT (top) and PDTB (bottom).

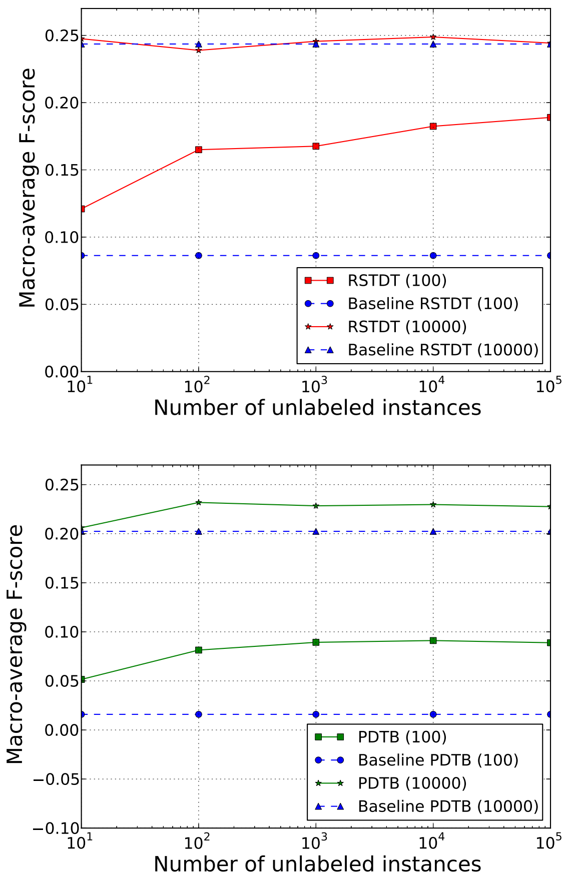


Figure 6: Macro-average F-score for RSTDT (top) and PDTB (bottom), for 100 and 10000 training instances, against the number of unlabeled instances.

ing unlabeled data is more obvious when the number of labeled training instances is small. In particular, with 100 training instances, the maximum improvement in F-score is 459% (corresponds to 100000 unlabeled instances). However, with 10000 labeled training instances the maximum improvement in F-score is 15% (corresponds to 100 unlabeled instances). These results confirm that, on the one hand performance improvement is more prominent for smaller training sets, and that on the other hand, performance is increased when using larger amounts of unlabeled data.

5 Conclusion

We presented a semi-supervised method which exploits the co-occurrence of features in unlabeled data, to extend feature vectors during training and testing in a discourse relation classifier. Despite the simplicity of the proposed method, it significantly improved the macro-average F-score in discourse relation classification for small training datasets, containing low-occurrence relations. We performed an evaluation on two popular datasets, the RSTDT and PDTB. We empirically evaluated the benefit of using a variable amount of unlabeled data for the proposed method. Although the macro-average F-scores of the classifiers described are too low to be used directly as discourse analyzers, the gain in F-score and accuracy for small labeled datasets are a promising perspective for improving classification accuracy for infrequent relation types. In particular, the proposed method can be employed in existing discourse classifiers that work well on popular relations, and be expected to improve the overall accuracy.

References

R. Basili, M. Cammisa, and A. Moschitti. 2006. A semantic kernel to classify texts with very few training examples. *Informatica (Slovenia)*, 30(2):163–172.

S. Bloehdorn, R. Basili, M. Cammisa, and A. Moschitti. 2006. Semantic kernels for text classification based on topological measures of feature similarity. In *Proc. of ICDM'06*, pages 808–812.

L. Carlson, D. Marcu, and M. E. Okurowski. 2001. Building a discourse-tagged corpus in the framework of Rhetorical Structure Theory. *Proc. of Second SIGdial Workshop on Discourse and Dialogue-Volume 16*, pages 1–10.

N. Cristianini, J. Shawe-Taylor, and H. Lodhi. 2002. Latent semantic kernels. *Journal of Intelligent Information Systems*, 18:127–152.

S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407.

D. A. duVerle and H. Prendinger. 2009. A novel discourse parser based on Support Vector Machine classification. In *Proc. of ACL'09*, pages 665–673.

H. Fang. 2008. A re-examination of query expansion using lexical resources. In *Proc. of ACL'08*, pages 139–147.

C. Fellbaum, editor. 1998. *WordNet: An electronic lexical database*. MIT Press.

H. Hernault, D. Bollegala, and M. Ishizuka. 2010. A sequential model for discourse segmentation. In *Proc. of CICLing'10*, pages 315–326.

D. Klein and C. D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems*, volume 15. MIT Press.

Z. Lin, M-Y. Kan, and H. T. Ng. 2009. Recognizing implicit discourse relations in the Penn Discourse Treebank. In *Proc. of EMNLP'09*, pages 343–351.

E. Loper and S. Bird. 2002. NLTK: The natural language toolkit. In *Proc. of ACL'02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics*, pages 63–70.

D. M. Magerman. 1995. Statistical decision-tree models for parsing. *Proc. of ACL'95*, pages 276–283.

W. C. Mann and S. A. Thompson. 1988. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

C. D. Manning and H. Schütze. 1999. *Foundations of Statistical Natural Language processing*. MIT Press.

D. Marcu and A. Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *Proc. of ACL'02*, pages 368–375.

D. Marcu. 2000. *The Theory and Practice of Discourse Parsing and Summarization*. MIT Press.

M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

N. Okazaki. 2009. Classias: A collection of machine-learning algorithms for classification. `http://www.chokkan.org/software/classias/`.

B. Pang and L. Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

E. Pitler, M. Raghupathy, H. Mehta, A. Nenkova, A. Lee, and A. Joshi. 2008. Easily identifiable discourse relations. In *Proc. of COLING'08 (Posters)*, pages 87–90.

E. Pitler, A. Louis, and A. Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proc. of ACL'09*, pages 683–691.

P. Piwek, H. Hernault, H. Prendinger, and M. Ishizuka. 2007. Generating dialogues between virtual agents automatically from text. In *Proc. of IVA'07*, pages 161–174.

R. L. Plackett. 1983. Karl Pearson and the chi-squared test. *International Statistical Review / Revue Internationale de Statistique*, 51(1):59–72.

R. Prasad, N. Dinesh, A. Lee, E. Miltsakaki, L. Robaldo, A. Joshi, and B. Webber. 2008a. The Penn Discourse TreeBank 2.0. In *Proc. of LREC'08*.

R. Prasad, E. Miltsakaki, N. Dinesh, A. Lee, A. Joshi, L. Robaldo, and B. Webber. 2008b. The Penn Discourse Treebank 2.0 annotation manual. Technical report, University of Pennsylvania Institute for Research in Cognitive Science.

G. Salton and C. Buckley. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company.

G. Siolas and F. d'Alché-Buc. 2000. Support Vector Machines based on a semantic kernel for text categorization. In *Proc. of IJCNN'00*, volume 5, page 5205.

S. Somasundaran, G. Namata, J. Wiebe, and L. Getoor. 2009. Supervised and unsupervised methods in employing discourse relations for improving opinion polarity classification. In *Proc. of EMNLP'09*, pages 170–179.

R. Soricut and D. Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. *Proc. of NA-ACL'03*, 1:149–156.

V. N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc.

A Game-Theoretic Approach to Generating Spatial Descriptions

Dave Golland
UC Berkeley
Berkeley, CA 94720
dsg@cs.berkeley.edu

Percy Liang
UC Berkeley
Berkeley, CA 94720
pliang@cs.berkeley.edu

Dan Klein
UC Berkeley
Berkeley, CA 94720
klein@cs.berkeley.edu

Abstract

Language is sensitive to both semantic and pragmatic effects. To capture both effects, we model language use as a cooperative game between two players: a speaker, who generates an utterance, and a listener, who responds with an action. Specifically, we consider the task of generating spatial references to objects, wherein the listener must accurately identify an object described by the speaker. We show that a speaker model that acts optimally with respect to an explicit, embedded listener model substantially outperforms one that is trained to directly generate spatial descriptions.

1 Introduction

Language is about successful communication between a speaker and a listener. For example, if the goal is to reference the target object O1 in Figure 1, a speaker might choose one of the following two utterances:

(a) *right of* O2 $\qquad (b)$ *on* O3

Although both utterances are semantically correct, (a) is ambiguous between O1 and O3, whereas (b) unambiguously identifies O1 as the target object, and should therefore be preferred over (a). In this paper, we present a game-theoretic model that captures this communication-oriented aspect of language interpretation and generation.

Successful communication can be broken down into semantics and pragmatics. Most computational work on interpreting language focuses on compositional semantics (Zettlemoyer and Collins, 2005; Wong and Mooney, 2007; Piantadosi et al., 2008), which is concerned with verifying the truth of a sentence. However, what is missing from this truth-oriented view is the pragmatic aspect of language—that language is used to accomplish an end goal, as exemplified by speech acts (Austin, 1962). Indeed, although both utterances (a) and (b) are semantically valid, only (b) is pragmatically felicitous: (a) is ambiguous and therefore violates the Gricean maxim of manner (Grice, 1975). To capture this maxim, we develop a model of pragmatics based on game theory, in the spirit of Jäger (2008) but extended to the stochastic setting. We show that Gricean maxims

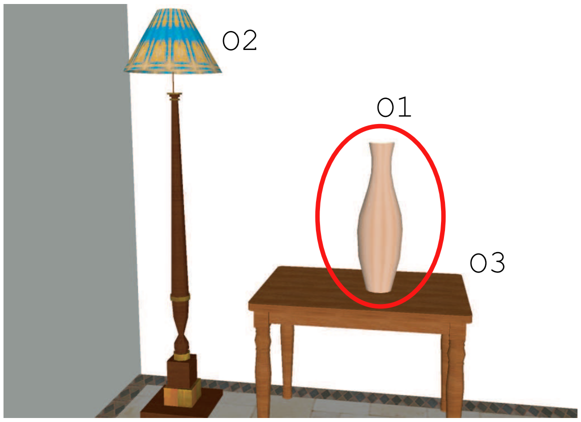


Figure 1: An example of a 3D model of a room. The *speaker*'s goal is to reference the target object O1 by describing its spatial relationship to other object(s). The *listener*'s goal is to guess the object given the speaker's description.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 410–419,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

fall out naturally as consequences of the model.

An effective way to empirically explore the pragmatic aspects of language is to work in the grounded setting, where the basic idea is to map language to some representation of the non-linguistic world (Yu and Ballard, 2004; Feldman and Narayanan, 2004; Fleischman and Roy, 2007; Chen and Mooney, 2008; Frank et al., 2009; Liang et al., 2009). Along similar lines, past work has also focused on interpreting natural language instructions (Branavan et al., 2009; Eisenstein et al., 2009; Kollar et al., 2010), which takes into account the goal of the communication. This work differs from ours in that it does not clarify the formal relationship between pragmatics and the interpretation task. Pragmatics has also been studied in the context of dialog systems. For instance, DeVault and Stone (2007) present a model of collaborative language between multiple agents that takes into account contextual ambiguities.

We present our pragmatic model in a grounded setting where a speaker must describe a target object to a listener via spatial description (such as in the example given above). Though we use some of the techniques from work on the semantics of spatial descriptions (Regier and Carlson, 2001; Gorniak and Roy, 2004; Tellex and Roy, 2009), we empirically demonstrate that having a model of pragmatics enables more successful communication.

2 Language as a Game

To model Grice's cooperative principle (Grice, 1975), we formulate the interaction between a speaker S and a listener L as a cooperative game, that is, one in which S and L share the same utility function. For simplicity, we focus on the production and interpretation of single utterances, where the speaker and listener have access to a shared context. To simplify notation, we suppress writing the dependence on the context.

The Communication Game

1. In order to communicate a *target* o to L, S produces an *utterance* w chosen according to a strategy $p_{\text{S}}(w \mid o)$.
2. L interprets w and responds with a *guess* g according to a strategy $p_{\text{L}}(g \mid w)$.
3. S and L collectively get a utility of $U(o, g)$.

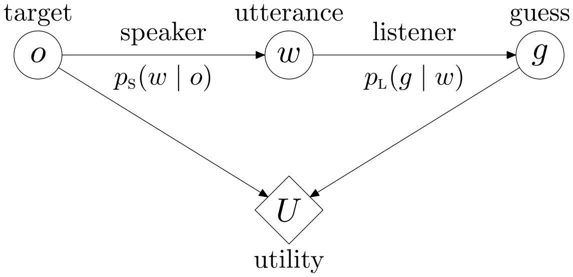


Figure 2: Diagram representing the communication game. A target, o, is given to the speaker that generates an utterance w. Based on this utterance, the listener generates a guess g. If $o = g$, then both the listener and speaker get a utility of 1, otherwise they get a utility of 0.

This communication game is described graphically in Figure 2. Figure 3 shows several instances of the communication game being played for the scenario in Figure 1.

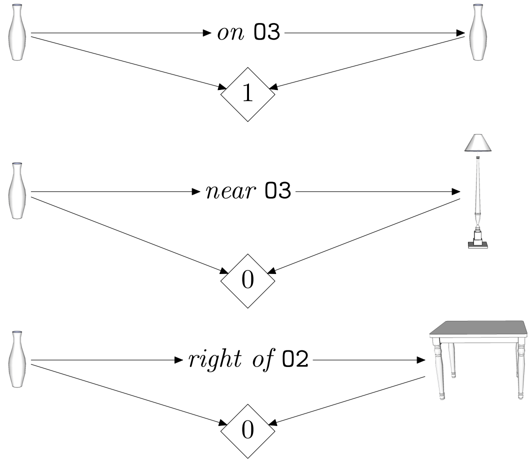


Figure 3: Three instances of the communication game on the scenario in Figure 1. For each instance, the target o, utterance w, guess g, and the resulting utility U are shown in their respective positions. A utility of 1 is awarded only when the guess matches the target.

Grice's maxim of manner encourages utterances to be unambiguous, which motivates the following utility, which we call *(communicative) success*:

$$U(o, g) \overset{\text{def}}{=} \mathbb{I}[o = g], \qquad (1)$$

where the indicator function $\mathbb{I}[o = g]$ is 1 if $o = g$ and 0 otherwise. Hence, a utility-maximizing speaker will attempt to produce unambiguous utterances because they increase the probability that the listener will correctly guess the target.

Given a speaker strategy $p_{\text{S}}(w \mid o)$, a listener strategy $p_{\text{L}}(g \mid w)$, and a prior distribution over targets $p(o)$, the expected utility obtained by S and L is as follows:

$$\begin{aligned} \text{EU}(\text{S},\text{L}) &= \sum_{o,w,g} p(o) p_{\text{S}}(w|o) p_{\text{L}}(g|w) U(o,g) \\ &= \sum_{o,w} p(o) p_{\text{S}}(w|o) p_{\text{L}}(o|w). \qquad (2) \end{aligned}$$

3 From Reflex Speaker to Rational Speaker

Having formalized the language game, we now explore various speaker and listener strategies. First, let us consider *literal* strategies. A literal speaker (denoted S:LITERAL) chooses uniformly from the set of utterances consistent with a target object, i.e., the ones which are semantically valid;[1] a literal listener (denoted L:LITERAL) guesses an object consistent with the utterance uniformly at random.

In the running example (Figure 1), where the target object is O1, there are two semantically valid utterances:

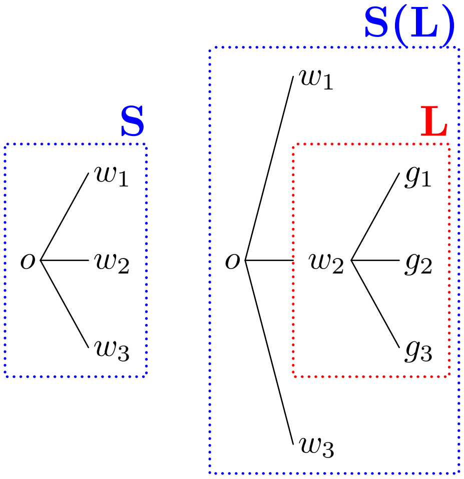
(a) *right of* O2 $\qquad (b)$ *on* O3

S:LITERAL selects (a) or (b) each with probability $\frac{1}{2}$. If S:LITERAL chooses (a), L:LITERAL will guess the target object O1 correctly with probability $\frac{1}{2}$; if S:LITERAL chooses (b), L:LITERAL will guess correctly with probability 1. Therefore, the expected utility EU(S:LITERAL, L:LITERAL) $= \frac{3}{4}$.

We say S:LITERAL is an example of a *reflex* speaker because it chooses an utterance without taking the listener into account. A general reflex speaker is depicted in Figure 4(a), where each edge represents a potential utterance.

Suppose we now have a model of some listener L. Motivated by game theory, we would optimize the expected utility (2) given $p_{\text{L}}(g \mid w)$. We call the resulting speaker S(L) the *rational* speaker with respect to listener L. Solving for this strategy yields:

$$\begin{aligned} p_{\text{S(L)}}(w \mid o) &= \mathbb{I}[w = w^*], \text{ where} \\ w^* &= \operatorname*{argmax}_{w'} p_{\text{L}}(o \mid w'). \qquad (3) \end{aligned}$$



(a) Reflex speaker (b) Rational speaker

Figure 4: (a) A reflex speaker (S) directly selects an utterance based only on the target object. Each edge represents a different choice of utterance. (b) A rational speaker (S(L)) selects an utterance based on an embedded model of the listener (L). Each edge in the first layer represents a different choice the speaker can make, and each edge in the second layer represents a response of the listener.

Intuitively, S(L) chooses an utterance, w^*, such that, if listener L were to interpret w^*, the probability of L guessing the target would be maximized.[2] The rational speaker is depicted in Figure 4(b), where, as before, each edge at the first level represents a possible choice for the speaker, but there is now a second layer representing the response of the listener.

To see how an embedded model of the listener improves communication, again consider our running example in Figure 1. A speaker can describe the target object O1 using either w_1 = *on* O3 or w_2 = *right of* O2. Suppose the embedded listener is L:LITERAL, which chooses uniformly from the set of objects consistent with the given utterance. In this scenario, $p_{\text{L:LITERAL}}(\text{O1} \mid w_1) = 1$ because w_1 unambiguously describes the target object, but $p_{\text{L:LITERAL}}(\text{O1} \mid w_2) = \frac{1}{2}$. The rational speaker S(L:LITERAL) would therefore choose w_1, achieving a utility of 1, which is an improvement over the reflex speaker S:LITERAL's utility of $\frac{3}{4}$.

[1] Semantic validity is approximated by a set of heuristic rules (e.g. *left* is all positions with smaller x-coordinates).

[2] If there are ties, any distribution over the utterances having the same utility is optimal.

4 From Literal Speaker to Learned Speaker

In the previous section, we showed that a literal strategy, one that considers only semantically valid choices, can be used to directly construct a reflex speaker S:LITERAL or an embedded listener in a rational speaker S(L:LITERAL). This section focuses on an orthogonal direction: improving literal strategies with learning. Specifically, we construct learned strategies from log-linear models trained on human annotations. These learned strategies can then be used to construct reflex and rational speaker variants—S:LEARNED and S(L:LEARNED), respectively.

4.1 Training a Log-Linear Speaker/Listener

We train the speaker, S:LEARNED, (similarly, listener, L:LEARNED) on training examples which comprise the utterances produced by the human annotators (see Section 6.1 for details on how this data was collected). Each example consists of a 3D model of a room in a house that specifies the 3D positions of each object and the coordinates of a 3D camera. When training the speaker, each example is a pair (o, w), where o is the input target object and w is the output utterance. When training the listener, each example is (w, g), where w is the input utterance and g is the output guessed object.

For now, an utterance w consists of two parts:

- A spatial preposition $w.r$ (e.g., *right of*) from a set of possible prepositions.[3]
- A reference object $w.o$ (e.g., `O3`) from the set of objects in the room.

We consider more complex utterances in Section 5.

Both S:LEARNED and L:LEARNED are parametrized by log-linear models:

$$p_{\text{S:LEARNED}}(w|o;\theta_{\text{S}}) \propto \exp\{\theta_{\text{S}}^\top \phi(o,w)\} \quad (4)$$

$$p_{\text{L:LEARNED}}(g|w;\theta_{\text{L}}) \propto \exp\{\theta_{\text{L}}^\top \phi(g,w)\} \quad (5)$$

where $\phi(\cdot,\cdot)$ is the feature vector (see below), θ_{S} and θ_{L} are the parameter vectors for speaker and listener. Note that the speaker and listener use the same set of features, but they have different parameters. Furthermore, the first normalization sums over possible utterances w while the second normalization sums over possible objects g in the scene. The two parameter vectors are trained to optimize the log-likelihood of the training data under the respective models.

Features We now describe the features $\phi(o, w)$. These features draw inspiration from Landau and Jackendoff (1993) and Tellex and Roy (2009).

Each object o in the 3D scene is represented by its bounding box, which is the smallest rectangular prism containing o. The following are functions of the camera, target (or guessed object) o, and the reference object $w.o$ in the utterance. The full set of features is obtained by conjoining these functions with indicator functions of the form $\mathbb{I}[w.r = r]$, where r ranges over the set of valid prepositions.

- *Proximity functions* measure the distance between o and $w.o$. This is implemented as the minimum over all the pairwise Euclidean distances between the corners of the bounding boxes. We also have indicator functions for whether o is the closest object, among the top 5 closest objects, and among the top 10 closest objects to $w.o$.
- *Topological functions* measure containment between o and $w.o$: $vol(o \cap w.o)/vol(o)$ and $vol(o \cap w.o)/vol(w.o)$. To simplify volume computation, we approximate each object by a bounding box that is aligned with the camera axes.
- *Projection functions* measure the relative position of the bounding boxes with respect to one another. Specifically, let v be the vector from the center of $w.o$ to the center of o. There is a function for the projection of v onto each of the axes defined by the camera orientation (see Figure 5). Additionally, there is a set of indicator functions that capture the relative magnitude of these projections. For example, there is a indicator function denoting whether the projection of v onto the camera's x-axis is the largest of all three projections.

[3] We chose 10 prepositions commonly used by people to describe objects in a preliminary data gathering experiment. This list includes multi-word units, which function equivalently to prepositions, such as *left of*.

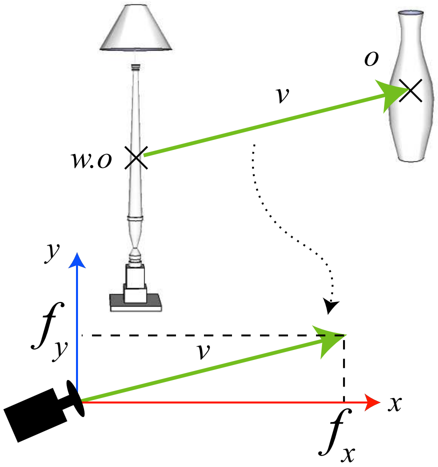


Figure 5: The projection features are computed by projecting a vector v extending from the center of the reference object to the center of the target object onto the camera axes f_x and f_y.

5 Handling Complex Utterances

So far, we have only considered speakers and listeners that deal with utterances consisting of one preposition and one reference object. We now extend these strategies to handle more complex utterances. Specifically, we consider utterances that conform to the following grammar:[4]

[noun]	N	→	*something* | O1 | O2 | · · ·
[relation]	R	→	*in front of* | *on* | · · ·
[conjunction]	NP	→	N RP*
[relativization]	RP	→	R NP

This grammar captures two phenomena of language use, conjunction and relativization.

- Conjunction is useful when one spatial relation is insufficient to disambiguate the target object. For example, in Figure 1, *right of* O2 could refer to the vase or the table, but using the conjunction *right of* O2 *and on* O3 narrows down the target object to just the vase.
- The main purpose of relativization is to refer to objects without a precise nominal descriptor. With complex utterances, it is possible to chain relative prepositional phrases, for example, using *on something right of* O2 to refer to the vase.

Given an utterance w, we define its *complexity* $|w|$ as the number of applications of the relativization rule, RP → R NP, used to produce w. We had only considered utterances of complexity 1 in previous sections.

5.1 Example Utterances

To illustrate the types of utterances available under the grammar, again consider the scene in Figure 1.

Utterances of complexity 2 can be generated either using the relativization rule exclusively, or both the conjunction and relativization rules. The relativization rule can be used to generate the following utterances:

- *on something that is right of* O2
- *right of something that is left of* O3

Applying the conjunction rule leads to the following utterances:

- *right of* O2 *and on* O3
- *right of* O2 *and under* O1
- *left of* O1 *and left of* O3

Note that we inserted the words *that is* after each N and the word *and* between every adjacent pair of RPs generated via the conjunction rule. This is to help a human listener interpret an utterance.

5.2 Extending the Rational Speaker

Suppose we have a rational speaker S(L) defined in terms of an embedded listener L which operates over utterances of complexity 1. We first extend L to interpret arbitrary utterances of our grammar. The rational speaker (defined in (2)) automatically inherits this extension.

Compositional semantics allows us to define the interpretation of complex utterances in terms of simpler ones. Specifically, each node in the parse tree has a *denotation*, which is computed recursively in terms of the node's children via a set of simple rules. Usually, denotations are represented as lambda-calculus functions, but for us, they will be distributions over objects in the scene. As a base case for interpreting utterances of complexity 1, we can use either L:LITERAL or L:LEARNED (defined in Sections 3 and 4).

[4] Naturally, we disallow direct reference to the target object.

Given a subtree w rooted at $u \in \{\texttt{N}, \texttt{NP}, \texttt{RP}\}$, we define the denotation of w, $[\![w]\!]$, to be a distribution over the objects in the scene in which the utterance was generated. The listener strategy $p_{\text{L}}(g|w) = [\![w]\!]$ is recursively as follows:

- If w is rooted at N with a single child x, then $[\![w]\!]$ is the uniform distribution over $\mathcal{N}(x)$, the set of objects consistent with the word x.
- If w is rooted at NP, we recursively compute the distributions over objects g for each child tree, multiply the probabilities, and renormalize (Hinton, 1999).
- If w is rooted at RP with relation r, we recursively compute the distribution over objects g' for the child NP tree. We then appeal to the base case to produce a distribution over objects g which are related to g' via relation r.

This strategy is defined formally as follows:

$$p_{\text{L}}(g \mid w) \propto \begin{cases} \mathbb{I}[g \in \mathcal{N}(x)] & w = (\texttt{N}\, x) \\ \prod_{j=1}^{k} p_{\text{L}}(g \mid w_j) & w = (\texttt{NP}\, w_1 \dots w_k) \\ \sum_{g'} p_{\text{L}}(g \mid (r, g')) p_{\text{L}}(g' \mid w') & w = (\texttt{RP}\, (\texttt{R}\, r)\, w') \end{cases} \tag{6}$$

Figure 6 shows an example of this bottom-up denotation computation for the utterance *on something right of* O2 with respect to the scene in Figure 1. The denotation starts with the lowest NP node $[\![\texttt{O2}]\!]$, which places all the mass on O2 in the scene. Moving up the tree, we compute the denotation of the RP, $[\![$*right of* O2$]\!]$, using the RP case of (6), which results in a distribution that places equal mass on O1 and O3.[5] The denotation of the N node $[\![$*something*$]\!]$ is a flat distribution over all the objects in the scene. Continuing up the tree, the denotation of the NP is computed by taking a product of the object distributions, and turns out to be exactly the same split distribution as its RP child. Finally, the denotation at the root is computed by applying the base case to *on* and the resulting distribution from the previous step.

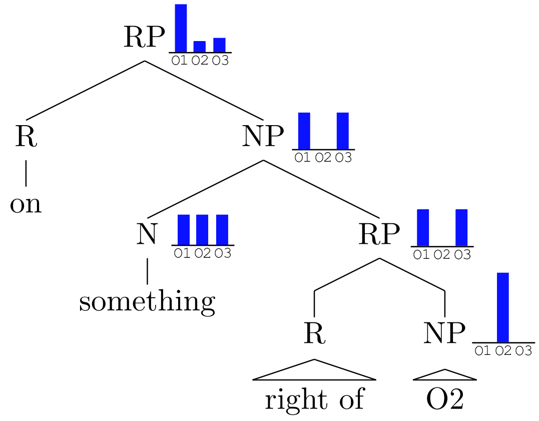


Figure 6: The listener model maps an utterance to a distribution over objects in the room. Each internal NP or RP node is a distribution over objects in the room.

Generation So far, we have defined the listener strategy $p_{\text{L}}(g \mid w)$. Given target o, the rational speaker S(L) with respect to this listener needs to compute $\text{argmax}_w\, p_{\text{L}}(o \mid w)$ as dictated by (3). This maximization is performed by enumerating all utterances of bounded complexity.

5.3 Modeling Listener Confusion

One shortcoming of the previous approach for extending a listener is that it falsely assumes that a listener can reliably interpret a simple utterance just as well as it can a complex utterance.

We now describe a more realistic speaker which is robust to listener confusion. Let $\alpha \in [0, 1]$ be a *focus parameter* which determines the confusion level. Suppose we have a listener L. When presented with an utterance w, for each application of the relativization rule, we have a $1 - \alpha$ probability of losing focus. If we stay focused for the entire utterance (with probability $\alpha^{|w|}$), then we interpret the utterance according to p_{L}. Otherwise (with probability $1 - \alpha^{|w|}$), we guess an object at random according to $p_{\text{rnd}}(g \mid w)$. We then use (3) to define the rational speaker S(L) with respect the following "confused listener" strategy:

$$\tilde{p}_{\text{L}}(g \mid w) = \alpha^{|w|} p_{\text{L}}(g \mid w) + (1 - \alpha^{|w|}) p_{\text{rnd}}(g \mid w). \tag{7}$$

As $\alpha \to 0$, the confused listener is more likely to make a random guess, and thus there is a stronger penalty against using more complex utterances. As

[5] It is worth mentioning that this split distribution between O1 and O3 represents the ambiguity mentioned in Section 3 when discussing the shortcomings of S:LITERAL.

$\alpha \rightarrow 1$, the confused listener converges to p_{L} and the penalty for using complex utterances vanishes.

5.4 The Taboo Setting

Notice that the rational speaker as defined so far does not make full use of our grammar. Specifically, the rational speaker will never use the "wildcard" noun *something* nor the relativization rule in the grammar because an NP headed by the wildcard *something* can always be replaced by the object ID to obtain a higher utility. For instance, in Figure 6, the NP spanning *something right of* O2 can be replaced by O3.

However, it is not realistic to assume that all objects can be referenced directly. To simulate scenarios where some objects cannot be referenced directly (and to fully exercise our grammar), we introduce the *taboo setting*. In this setting, we remove from the lexicon some fraction of the object IDs which are closest to the target object. Since the tabooed objects cannot be referenced directly, a speaker must resort to use of the wildcard *something* and relativization.

For example, in Figure 7, we enable tabooing around the target O1. This prevents the speaker from referring directly to O3, so the speaker is forced to describe O3 via the relativization rule, for example, producing *something right of* O2.

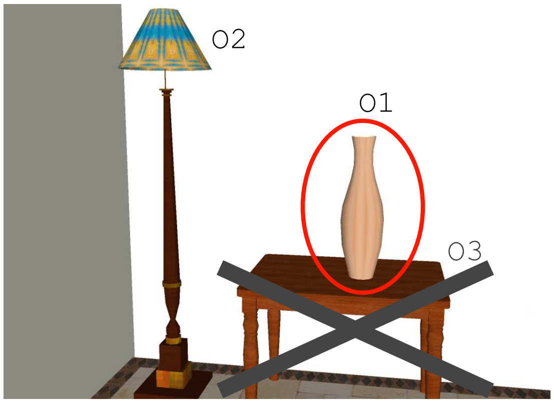


Figure 7: With tabooing enabled around O1, O3 can no longer be referred to directly (represented by an X).

6 Experiments

We now present our empirical results, showing that rational speakers, who have embedded models of listeners, can communicate more successfully than reflex speakers, who do not.

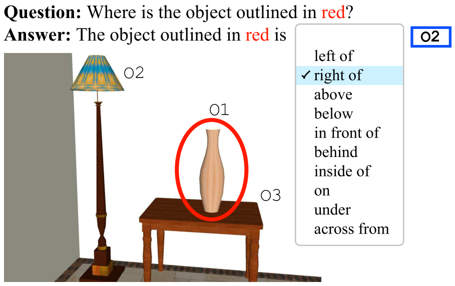

Question: Where is the object outlined in red?
Answer: The object outlined in red is

Figure 8: Mechanical Turk speaker task: Given the target object (e.g., O1), a human speaker must choose an utterance to describe the object (e.g., *right of* O2).

6.1 Setup

We collected 43 scenes (rooms) from the Google Sketchup 3D Warehouse, each containing an average of 22 objects (household items and pieces of furniture arranged in a natural configuration). For each object o in a scene, we create a *scenario*, which represents an instance of the communication game with o as the target object. There are a total of 2,860 scenarios, which we split evenly into a training set (denoted TR) and a test set (denoted TS).

We created the following two Amazon Mechanical Turk tasks, which enable humans to play the language game on the scenarios:

Speaker Task In this task, human annotators play the role of speakers in the language game. They are prompted with a target object o and asked to each produce an utterance w (by selecting a preposition $w.r$ from a dropdown list and clicking on a reference object $w.o$) that best informs a listener of the identity of the target object.

For each training scenario o, we asked three speakers to produce an utterance w. The three resulting (o, w) pairs are used to train the learned reflex speaker (S:LITERAL). These pairs were also used to train the learned reflex listener (L:LITERAL), where the target o is treated as the guessed object. See Section 4.1 for the details of the training procedure.

Listener Task In this task, human annotators play the role of listeners. Given an utterance generated by a speaker (human or not), the human listener must

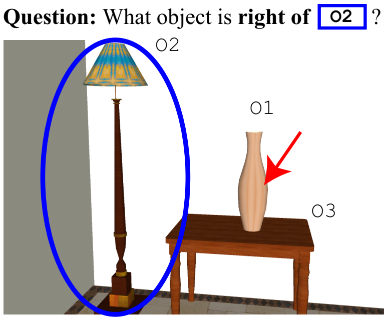


Figure 9: Mechanical Turk listener task: a human listener is prompted with an utterance generated by a speaker (e.g., *right of* O2), and asked to click on an object (shown by the red arrow).

guess the target object that the speaker saw by clicking on an object. The purpose of the listener task is to evaluate speakers, as described in the next section.

6.2 Evaluation

Utility (Communicative Success) We primarily evaluate a speaker by its ability to communicate successfully with a human listener. For each test scenario, we asked three listeners to guess the object. We use $p_{\text{L:HUMAN}}(g \mid w)$ to denote the distribution over guessed objects g given prompt w. For example, if two of the three listeners guessed O1, then $p_{\text{L:HUMAN}}(\texttt{O1} \mid w) = \frac{2}{3}$. The expected utility (2) is then computed by averaging the utility (communicative success) over the test scenarios TS:

$$\begin{aligned}\text{SUCCESS}(\text{S}) &= \text{EU}(\text{S}, \text{L:HUMAN}) \qquad (8)\\ &= \frac{1}{|\text{TS}|}\sum_{o\in\text{TS}}\sum_{w} p_{\text{S}}(w|o)p_{\text{L:HUMAN}}(o|w).\end{aligned}$$

Exact Match As a secondary evaluation metric, we also measure the ability of our speaker to exactly match an utterance produced by a human speaker. Note that since there are many ways of describing an object, exact match is neither necessary nor sufficient for successful communication.

We asked three human speakers to each produce an utterance w given a target o. We use $p_{\text{S:HUMAN}}(w \mid o)$ to denote this distribution; for example, $p_{\text{S:HUMAN}}(\textit{right of}\ \texttt{O2} \mid o) = \frac{1}{3}$ if exactly one of the three speakers uttered *right of* O2. We then define the *exact match* of a speaker S as follows:

Speaker	Success	Exact Match
S:LITERAL [reflex]	4.62%	1.11%
S(L:LITERAL) [rational]	33.65%	2.91%
S:LEARNED [reflex]	38.36%	5.44%
S(L:LEARNED) [rational]	**52.63%**	14.03%
S:HUMAN	41.41%	**19.95%**

Table 1: Comparison of various speakers on communicative success and exact match, where only utterances of complexity 1 are allowed. The rational speakers (with respect to both the literal listener L:LITERAL and the learned listener L:LEARNED) perform better than their reflex counterparts. While the human speaker (composed of three people) has higher exact match (it is better at mimicking itself), the rational speaker S(L:LEARNED) actually achieves higher communicative success than the human listener.

$$\text{MATCH}(\text{S}) = \frac{1}{|\text{TS}|}\sum_{o\in\text{TS}}\sum_{w} p_{\text{S:HUMAN}}(w \mid o)p_{\text{S}}(w \mid o). \qquad (9)$$

6.3 Reflex versus Rational Speakers

We first evaluate speakers in the setting where only utterances of complexity 1 are allowed. Table 1 shows the results on both success and exact match. First, our main result is that the two rational speakers S(L:LITERAL) and S(L:LEARNED), which each model a listener explicitly, perform significantly better than the corresponding reflex speakers, both in terms of success and exact match.

Second, it is natural that the speakers that involve learning (S:LITERAL and S(L:LITERAL)) outperform the speakers that only consider the literal meaning of utterances (S:LEARNED and S(L:LEARNED)), as the former models capture subtler preferences using features.

Finally, we see that in terms of exact match, the human speaker S:HUMAN performs the best (this is not surprising because human exact match is essentially the inter-annotator agreement), but in terms of communicative success, S(L:LEARNED) achieves a higher success rate than S:HUMAN, suggesting that the game-theoretic modeling undertaken by the rational speakers is effective for communication, which is ultimate goal of language.

Note that exact match is low even for the "human speaker", since there are often many equally good

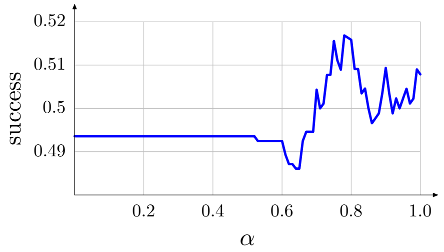


Figure 10: Communicative success as a function of focus parameter α without tabooing on TsDev. The optimal value of α is obtained at 0.79.

ways to evoke an object. At the same time, the success rates for all speakers are rather low, reflecting the fundamental difficulty of the setting: sometimes it is impossible to unambiguously evoke the target object via short utterances. In the next section, we show that we can improve the success rate by allowing the speakers to generate more complex utterances.

6.4 Generating More Complex Utterances

We now evaluate the rational speaker s(L:Learned) when it is allowed to generate utterances of complexity 1 or 2. Recall from Section 5.3 that the speaker depends on a focus parameter α, which governs the embedded listener's ability to interpret the utterance. We divided the test set (Ts) in two halves: TsDev, which we used to tune the value of α and TsFinal, which we used to evaluate success rates.

Figure 10 shows the communicative success as a function of α on TsDev. When α is small, the embedded listener is confused more easily by more complex utterances; therefore the speaker tends to choose mostly utterances of complexity 1. As α increases, the utterances increase in complexity, as does the success rate. However, when α approaches 1, the utterances are too complex and the success rate decreases. The dependence between α and average utterance complexity is shown in Figure 11.

Table 2 shows the success rates on TsFinal for $\alpha \to 0$ (all utterances have complexity 1), $\alpha = 1$ (all utterances have complexity 2), and α tuned to maximize the success rate based on TsDev. Setting α in this manner allows us to effectively balance complexity and ambiguity, resulting in an improvement in the success rate.

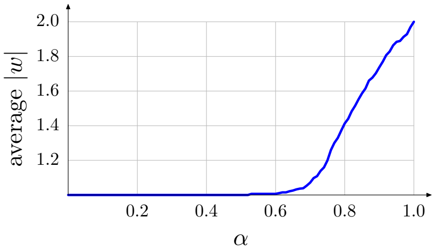


Figure 11: Average utterance complexity as a function of the focus parameter α on TsDev. Higher values of α yield more complex utterances.

Taboo Amount	Success ($\alpha \to 0$)	Success ($\alpha = 1$)	Success ($\alpha = \alpha^*$)	α^*
0%	51.78%	50.99%	54.53%	0.79
5%	38.75%	40.83%	43.12%	0.89
10%	29.57%	29.69%	30.30%	0.80
30%	12.40%	13.04%	12.98%	0.81

Table 2: Communicative success (on TsFinal) of the rational speaker s(L:Learned) for various values of α across different taboo amounts. When the taboo amount is small, small values of α lead to higher success rates. As the taboo amount increases, larger values of α (resulting in more complex utterances) are better.

7 Conclusion

Starting with the view that the purpose of language is successful communication, we developed a game-theoretic model in which a *rational* speaker generates utterances by explicitly taking the listener into account. On the task of generating spatial descriptions, we showed the rational speaker substantially outperforms a baseline reflex speaker that does not have an embedded model. Our results therefore suggest that a model of the pragmatics of communication is an important factor to consider for generation.

Acknowledgements This work was supported by the National Science Foundation through a Graduate Research Fellowship to the first two authors. We also would like to acknowledge Surya Murali, the designer of the 3D Google Sketchup models, and thank the anonymous reviewers for their comments.

References

J. L. Austin. 1962. *How to do Things with Words: The William James Lectures delivered at Harvard Univer-*

sity in 1955. Oxford, Clarendon, UK.

S. Branavan, H. Chen, L. S. Zettlemoyer, and R. Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, Singapore. Association for Computational Linguistics.

D. L. Chen and R. J. Mooney. 2008. Learning to sportscast: A test of grounded language acquisition. In *International Conference on Machine Learning (ICML)*, pages 128–135. Omnipress.

David DeVault and Matthew Stone. 2007. Managing ambiguities across utterances in dialogue.

J. Eisenstein, J. Clarke, D. Goldwasser, and D. Roth. 2009. Reading to learn: Constructing features from semantic abstracts. In *Empirical Methods in Natural Language Processing (EMNLP)*, Singapore.

J. Feldman and S. Narayanan. 2004. Embodied meaning in a neural theory of language. *Brain and Language*, 89:385–392.

M. Fleischman and D. Roy. 2007. Representing intentions in a cognitive model of language acquisition: Effects of phrase structure on situated verb learning. In *Association for the Advancement of Artificial Intelligence (AAAI)*, Cambridge, MA. MIT Press.

M. C. Frank, N. D. Goodman, and J. B. Tenenbaum. 2009. Using speakers' referential intentions to model early cross-situational word learning. *Psychological Science*, 20(5):578–585.

Peter Gorniak and Deb Roy. 2004. Grounded semantic composition for visual scenes. In *Journal of Artificial Intelligence Research*, volume 21, pages 429–470.

H. P. Grice. 1975. Syntax and Semantics; Logic and Conversation. 3:Speech Acts:41–58.

G. Hinton. 1999. Products of experts. In *International Conference on Artificial Neural Networks (ICANN)*.

G. Jäger. 2008. Game theory in semantics and pragmatics. Technical report, University of Tübingen.

T. Kollar, S. Tellex, D. Roy, and N. Roy. 2010. Toward understanding natural language directions. In *Human-Robot Interaction*, pages 259–266.

Barbara Landau and Ray Jackendoff. 1993. "what" and "where" in spatial language and spatial cognition. *Behavioral and Brain Sciences*, 16(2spatial prepositions analysis, cross linguistic conceptual similarities; comments/response):217–238.

P. Liang, M. I. Jordan, and D. Klein. 2009. Learning semantic correspondences with less supervision. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, Singapore. Association for Computational Linguistics.

S. T. Piantadosi, N. D. Goodman, B. A. Ellis, and J. B. Tenenbaum. 2008. A Bayesian model of the acquisition of compositional semantics. In *Proceedings of the Thirtieth Annual Conference of the Cognitive Science Society*.

T Regier and LA Carlson. 2001. Journal of experimental psychology. general; grounding spatial language in perception: an empirical and computational investigation. 130(2):273–298.

Stefanie Tellex and Deb Roy. 2009. Grounding spatial prepositions for video search. In *ICMI*.

Y. W. Wong and R. J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Association for Computational Linguistics (ACL)*, pages 960–967, Prague, Czech Republic. Association for Computational Linguistics.

C. Yu and D. H. Ballard. 2004. On the integration of grounding language and learning objects. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 488–493, Cambridge, MA. MIT Press.

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Uncertainty in Artificial Intelligence (UAI)*, pages 658–666.

Facilitating Translation Using Source Language Paraphrase Lattices

Jinhua Du, Jie Jiang, Andy Way
CNGL, School of Computing
Dublin City University, Dublin, Ireland
{jdu, jjiang, away}@computing.dcu.ie

Abstract

For resource-limited language pairs, coverage of the test set by the parallel corpus is an important factor that affects translation quality in two respects: 1) out of vocabulary words; 2) the same information in an input sentence can be expressed in different ways, while current phrase-based SMT systems cannot automatically select an alternative way to transfer the same information. Therefore, given limited data, in order to facilitate translation from the input side, this paper proposes a novel method to reduce the translation difficulty using source-side lattice-based paraphrases. We utilise the original phrases from the input sentence and the corresponding paraphrases to build a lattice with estimated weights for each edge to improve translation quality. Compared to the baseline system, our method achieves relative improvements of 7.07%, 6.78% and 3.63% in terms of BLEU score on small, medium and large-scale English-to-Chinese translation tasks respectively. The results show that the proposed method is effective not only for resource-limited language pairs, but also for resource-sufficient pairs to some extent.

1 Introduction

In recent years, statistical MT systems have been easy to develop due to the rapid explosion in data availability, especially parallel data. However, in reality there are still many language pairs which lack parallel data, such as Urdu–English, Chinese–Italian, where large amounts of speakers exist for both languages; of course, the problem is far worse for pairs such as Catalan–Irish. For such resource-limited language pairs, sparse amounts of parallel data would cause the word alignment to be inaccurate, which would in turn lead to an inaccurate phrase alignment, and bad translations would result. Callison-Burch et al. (2006) argue that limited amounts of parallel training data can lead to the problem of low coverage in that many phrases encountered at run-time are not observed in the training data and so their translations will not be learned. Thus, in recent years, research on addressing the problem of unknown words or phrases has become more and more evident for resource-limited language pairs.

Callison-Burch et al. (2006) proposed a novel method which substitutes a paraphrase for an unknown source word or phrase in the input sentence, and then proceeds to use the translation of that paraphrase in the production of the target-language result. Their experiments showed that by translating paraphrases a marked improvement was achieved in coverage and translation quality, especially in the case of unknown words which previously had been left untranslated. However, on a large-scale data set, they did not achieve improvements in terms of automatic evaluation.

Nakov (2008) proposed another way to use paraphrases in SMT. He generates nearly-equivalent syntactic paraphrases of the source-side training sentences, then pairs each paraphrased sentence with the target translation associated with the original sentence in the training data. Essentially, this method generates new training data using paraphrases to train a new model and obtain more useful

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 420–429,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

phrase pairs. However, he reported that this method results in bad system performance. By contrast, real improvements can be achieved by merging the phrase tables of the paraphrase model and the original model, giving priority to the latter. Schroeder et al. (2009) presented the use of word lattices for multi-source translation, in which the multiple source input texts are compiled into a compact lattice, over which a single decoding pass is then performed. This lattice-based method achieved positive results across all data conditions.

In this paper, we propose a novel method using paraphrases to facilitate translation, especially for resource-limited languages. Our method does not distinguish unknown words in the input sentence, but uses paraphrases of all possible words and phrases in the source input sentence to build a source-side lattice to provide a diverse and flexible list of source-side candidates to the SMT decoder so that it can search for a best path and deliver the translation with the highest probability. In this case, we neither need to change the phrase table, nor add new features in the log-linear model, nor add new sentences in the training data.

The remainder of this paper is organised as follows. In Section 2, we define the "translation difficulty" from the perspective of the source side, and then examine how well the test set is covered by the phrase table and the parallel training data . Section 3 describes our paraphrase lattice method and discusses how to set the weights for the edges in the lattice network. In Section 4, we report comparative experiments conducted on small, medium and large-scale English-to-Chinese data sets. In Section 5, we analyse the influence of our paraphrase lattice method. Section 6 concludes and gives avenues for future work.

2 What Makes Translation Difficult?

2.1 Translation Difficulty

We use the term "translation difficulty" to explain how difficult it is to translate the source-side sentence in three respects:

- The OOV rates of the source sentences in the test set (Callison-Burch et al., 2006).
- Translatability of a known phrase in the input sentence. Some particular grammatical structures on the source side cannot be directly translated into the corresponding structures on the target side. Nakov (2008) presents an example showing how hard it is to translate an English construction into Spanish. Assume that an English-to-Spanish SMT system has an entry in its phrase table for "*inequality of income*", but not for "*income inequality*". He argues that the latter phrase is hard to translate into Spanish where noun compounds are rare: the correct translation in this case requires a suitable Spanish preposition and a reordering, which are hard for the system to realize properly in the target language (Nakov, 2008).
- Consistency between the reference and the target-side sentence in the training corpus. Nakov (2008) points out that if the target-side sentence in the parallel corpus is inconsistent with the reference of the test set, then in some cases, a test sentence might contain pieces that are equivalent, but syntactically different from the phrases learned in training, which might result in practice in a missed opportunity for a high-quality translation. In this case, if we use paraphrases for these pieces of text, then we might improve the opportunity for the translation to approach the reference, especially in the case where only one reference is available.

2.2 Coverage

As to the first aspect – coverage – we argue that the coverage rate of the new words or unknown words are more and more becoming a "bottleneck" for resource-limited languages. Furthermore, current SMT systems, either phrase-based (Koehn et al., 2003; Chiang, 2005) or syntax-based (Zollmann and Venugopal, 2006), use phrases as the fundamental translation unit, so how much the phrase table and training data can cover the test set is an important factor which influences the translation quality. Table 1 shows the statistics of the coverage of the test set on English-to-Chinese FBIS data, where we can see that the coverage of unigrams is very high, especially when the data is increased to the medium size (200K), where unigram coverage is greater than 90%. Based on the observations of the unknown un-

PL	Tset	20K		Cov.(%)		200K		Cov.(%)	
		PT	Corpus	in PT	in Corpus	PT	Corpus	in PT	in Corpus
1	5,369	3,785	4,704	70.5	87.61	4,941	5,230	92.03	97.41
2	24,564	8,631	15,109	35.14	61.51	16,803	21,071	68.40	85.78
3	37,402	4,538	12,091	12.13	32.33	12,922	22,531	34.55	60.24
4	41,792	1,703	6,150	4.07	14.72	5,974	14,698	14.29	35.17
5	43,008	626	2,933	1.46	6.82	2,579	8,425	5.99	19.59
6	43,054	259	1,459	0.6	3.39	1,192	4,856	2.77	11.28
7	42,601	119	821	0.28	1.93	581	2,936	1.36	6.89
8	41,865	51	505	0.12	1.21	319	1,890	0.76	4.51
9	40,984	34	341	0.08	0.83	233	1,294	0.57	3.16
10	40,002	22	241	0.05	0.6	135	923	0.34	2.31

Table 1: The coverage of the test set by the phrase table and the parallel corpus based on different amount of the training data. "PL" indicates the Phrase Length N, where $\{1 <= N <= 10\}$; "20K" and "200K" represent the sizes of the parallel data for model training and phrase extraction; "Cov." indicates the coverage rate; "Tset" represents the number of unique phrases with the length N in the Test Set; "PT" represents the number of phrases of the Test Set occur in the Phrase Table; "Corpus" indicates the number of phrases of the Test Set appearing in the parallel corpus; "in PT" indicates the coverage of the phrases in the Test Set by the phrase table and correspondingly "in Corpus" represents the coverage of the phrases in the Test Set by the Parallel Corpus.

igrams, we found that most are named entities (NEs) such as person name, location name, etc. From the bigram phrases, the coverage rates begin to significantly decline. It can also be seen that phrases containing more than 5 words rarely appear either in the phrase table or in the parallel corpus, which indicates that data sparseness is severe for long phrases. Even if the size of the corpus is significantly increased (e.g. from 20K to 200K), the coverage of long phrases is still quite low.

With respect to these three aspects of the translation difficulty, especially for data-limited language pairs, we propose a more effective method to make use of the paraphrases to facilitate translation process.

3 Paraphrase Lattice for Input Sentences

In this Section, we propose a novel method to employ paraphrases to reduce the translation difficulty and in so doing increase the translation quality.

3.1 Motivation

Our idea to build a paraphrase lattice for SMT is inspired by the following points:

- Handling unknown words is a challenging issue for SMT, and using paraphrases is an effective way to facilitate this problem (Callison-Burch et al., 2006);
- The method of paraphrase substitution does not show any significant improvement, especially on a large-scale data set in terms of BLEU (Papineni et al., 2002) scores (Callison-Burch et al., 2006);
- Building a paraphrase lattice might provide more translation options to the decoder so that it can flexibly search for the best path.

The major contributions of our method are:

- We consider all N-gram phrases rather than only unknown phrases in the test set, where $\{1 <= N <= 10\}$;
- We utilise lattices rather than simple substitution to facilitate the translation process;
- We propose an empirical weight estimation method to set weights for edges in the word lattice, which is detailed in Section 3.4.

3.2 Paraphrase Acquisition

Paraphrases are alternative ways to express the same or similar meaning given a certain original word, phrase or segment. The paraphrases used in our method are generated from the parallel corpora based on the algorithm in (Bannard and Callison-Burch, 2005), in which paraphrases are identified by pivoting through phrases in another language. In this algorithm, the foreign language translations of an English phrase are identified, all occurrences of those foreign phrases are found, and all English phrases that they translate as are treated as potential paraphrases of the original English phrase (Callison-Burch et al., 2006). A paraphrase has a probability $p(e_2|e_1)$ which is defined as in (2):

$$p(e_2|e_1) = \sum_f p(f|e_1)p(e_2|f) \tag{1}$$

where the probability $p(f|e_1)$ is the probability that the original English phrase e_1 translates as a particular phrase f in the other language, and $p(e_2|f)$ is the probability that the candidate paraphrase e_2 translates as the foreign language phrase.

$p(e_2|f)$ and $p(f|e_1)$ are defined as the translation probabilities which can be calculated straightforwardly using maximum likelihood estimation by counting how often the phrases e and f are aligned in the parallel corpus as in (2) and (3):

$$p(e_2|f) \approx \frac{count(e_2, f)}{\sum_{e_2} count(e_2, f)} \tag{2}$$

$$p(f|e_1) \approx \frac{count(f, e_1)}{\sum_f count(f, e_1)} \tag{3}$$

3.3 Construction of Paraphrase Lattice

To present paraphrase options to the PB-SMT decoder, lattices with paraphrase options are constructed to enrich the source-language sentences. The construction process takes advantage of the correspondence between detected paraphrases and positions of the original words in the input sentence, then creates extra edges in the lattices to allow the decoder to consider paths involving the paraphrase words.

An toy example is illustrated in Figure 1: given a sequence of words $\{w_1, \ldots, w_N\}$ as the input, two phrases $\alpha = \{\alpha_1, \ldots, \alpha_p\}$ and $\beta = \{\beta_1, \ldots, \beta_q\}$ are detected as paraphrases for $S_1 = \{w_x, \ldots, w_y\}$ $(1 \leq x \leq y \leq N)$ and $S_2 = \{w_m, \ldots, w_n\}$ $(1 \leq m \leq n \leq N)$ respectively. The following steps are taken to transform them into word lattices:

1. Transform the original source sentence into word lattices. $N + 1$ nodes $(\theta_k, 0 \leq k \leq N)$ are created, and N edges (referred to as "ORG-E" edges) labeled with w_i $(1 \leq i \leq N)$ are generated to connect them sequentially.

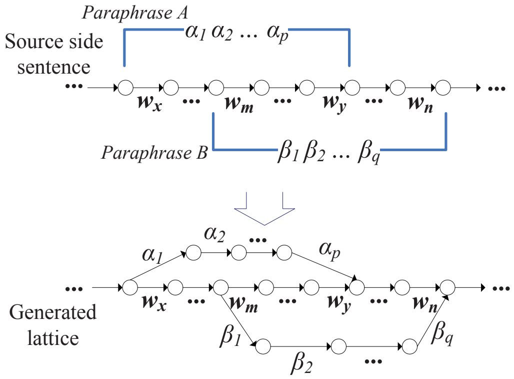


Figure 1: An example of lattice-based paraphrases for an input sentence

2. Generate extra nodes and edges for each of the paraphrases. Taking α as an example, firstly, $p - 1$ nodes are created, and then p edges (referred as "NEW-E" edges) labeled with α_j $(1 \leq j \leq p)$ are generated to connect node θ_{x-1}, $p - 1$ nodes and θ_{y-1}.

Via step 2, word lattices are generated by adding new nodes and edges coming from paraphrases. Note that to build word lattices, paraphrases with multi-words are broken into word sequences, and each of the words produces one extra edge in the word lattices as shown in the bottom part in Figure 1.

Figure 2 shows an example of constructing the word lattice for an input sentence which is from the test set used in our experiments.[1] The top part in Figure 2 represents nodes (double-line circles) and edges (solid lines) that are constructed by the original words from the input sentence, while the bottom part in Figure 2 indicates the final word lattice with the addition of new nodes (single-line circles) and new edges (dashed lines) which come from the paraphrases. We can see that the paraphrase lattice increases the diversity of the source phrases so that it can provide more flexible translation options during the decoding process.

[1] Figure 2 contains paths that are duplicates except for the weights. We plan to handle this in future work.

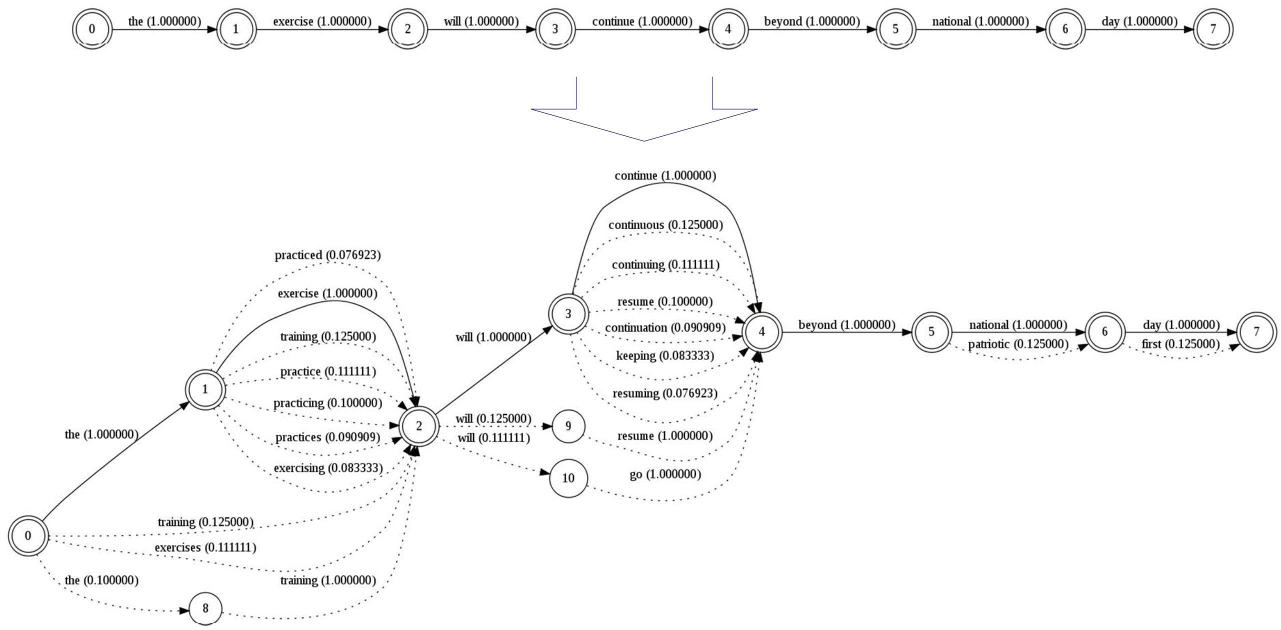


Figure 2: An example of how to build a paraphrase lattice for an input sentence

3.4 Weight Estimation

Estimating and normalising the weight for each edge in the word lattice is a challenging issue when the edges come from different sources. In this section, we propose an empirical method to set the weights for the edges by distinguishing the original ("ORG-E") and new ("NEW-E") edges in the lattices. The aim is to utilize the original sentences as the references to weight the edges from paraphrases, so that decoding paths going through "ORG-E" edges will tend to have higher scores than those which use "NEW-E" ones. The assumption behind this is that the paraphrases are alternatives for the original sentences, so decoding paths going though them ought to be penalised.

Therefore, for all the "ORG-E" edges, their weights in the lattice are set to 1.0 as the reference. Thus, in the log-linear model, decoding paths going though these edges are not penalised because they do not come from the paraphrases.

By contrast, "NEW-E" are divided into two groups for the calculation of weights:

- For "NEW-E" edges which are outgoing edges of the lattice nodes that come from the original sentences, the probabilities $p(e_s|e_i)^2$ of their corresponding paraphrases are utilised to produce empirical weights. Supposing that a set of paraphrases $\mathbf{X} = \{x_1, \ldots, x_k\}$ start at node A which comes from the original sentence, so that $\mathbf{X}$ are sorted descendingly based on the probabilities $p(e_s|e_i)$, their corresponding edges for node A are $\mathbf{G} = \{g_1, \ldots, g_k\}$, then the weights are calculated as in (4):

$$w(e_i) = \frac{1}{k+i} \;\; (1 <= i <= k) \qquad (4)$$

  where k is a predefined parameter to trade off between decoding speed and the number of potential paraphrases being considered. Thus, once a decoding path goes though one of these edges, it will be penalised according to its paraphrase probabilities.

- For all other "NEW-E" edges, their weights are set to 1.0, because the paraphrase penalty has been counted in their preceding "NEW-E" edges.

Figure 2 illustrates the weight estimation results. Nodes coming from the original sentences are drawn in double-line circles (e.g. nodes 0 to 7), while

[2] e_s indicates the source phrase S, e_i represents one of the paraphrases of S.

nodes created from paraphrases are shown in single-line circles (e.g. nodes 8 to 10). "ORG-E" edges are drawn in solid lines and "NEW-E" edges are shown using dashed lines. As specified previously, "ORG-E" edges are all weighted by 1.0 (e.g. edge labeled "the" from node 0 to 1). By contrast, "NEW-E" edges in the first group are weighted by equation (4) (e.g. edges in dashed lines start from node 0 to node 2 and 8), while others in the second group are weighted by 1.0 (e.g. edge labeled "training" from node 8 to 2). Note that penalties of the paths going through paraphrases are counted by equation (4), which is represented by the weights of "NEW-E" edges in the first group. For example, starting from node 2, paths going to node 9 and 10 are penalised because lattice weights are also considered in the log-linear model. However, other edges do not imply penalties since their weights are set to 1.0.

The reason to set all weights for the "ORG-E" edges to a uniform weight (e.g. 1.0) instead of a lower empirical weight is to avoid excessive penalties for the original words. For example, in Figure 2, the original edge from node 3 to 4 (*continue*) has a weight of 1.0, so the paths going though the original edges from node 2 to 4 (*will continue*) have a higher lattice score ($1.0 \times 1.0 = 1.0$) than the paths going through the edges of paraphrases (e.g. *will resume* (score: $0.125 \times 1.0 = 0.125$) and *will go* (score: $0.11 \times 1.0 = 0.11$)), or any other mixed paths that goes through original edges and paraphrase edges, such as *will continuous* (score: $1.0 \times 0.125 = 0.125$). The point is that we should have more trust when translating the original words, but if we penalise (set weights < 1.0) the "ORG-E" edges whenever there is a paraphrase for them, then when considering the context of the lattice, paraphrases will be favoured systematically. That is why we just penalise the "NEW-E" edges in the first group and set other weights to 1.0.

As to unknown words in the input sentence, even if we give them a prioritised weight, they would be severely penalised in the decoding process. So we do not need to distinguish unknown words when building and weighting the paraphrase lattice.

4 Experiments

4.1 System and Data Preparation

For our experiments, we use Moses (Koehn et al., 2007) as the baseline system which can support lattice decoding. We also realise a paraphrase substitution-based system (Para-Sub)[3] based on the method in (Callison-Burch, 2006) to compare with the baseline system and our proposed paraphrase lattice-based (Lattice) system.

The alignment is carried out by GIZA++ (Och and Ney, 2003) and then we symmetrized the word alignment using the grow-diag-final heuristic. The maximum phrase length is 10 words. Parameter tuning is performed using Minimum Error Rate Training (Och, 2003).

The experiments are conducted on English-to-Chinese translation. In order to fully compare our proposed method with the baseline and the "Para-Sub" system, we perform the experiments on three different sizes of training data: 20K, 200K and 2.1 million pairs of sentences. The former two sizes of data are derived from FBIS,[4] and the latter size of data consists of part of HK parallel corpus,[5] ISI parallel data,[6] other news data and parallel dictionaries from LDC. All the language models are 5-gram which are trained on the monolingual part of parallel data.

The development set (devset) and the test set for experiments using 20K and 200K data sets are randomly extracted from the FBIS data. Each set includes 1,200 sentences and each source sentence has one reference. For the 2.1 million data set, we use a different devset and test set in order to verify whether our proposed method can work on a language pair with sufficient resources. The devset is the NIST 2005 Chinese-English current set which has only one reference for each source sentence and the test set is the NIST 2003 English-to-Chinese current set which contains four references for each source sentence. All results are reported in BLEU and TER (Snover et al., 2006) scores.

[3] We use "Para-Sub" to represent their system in the rest of this paper.

[4] This is a multilingual paragraph-aligned corpus with LDC resource number LDC2003E14.

[5] LDC number: LDC2004T08.

[6] LDC number: LDC2007T09.

SYS	20K				200K			
	BLEU	CI 95%	pair-CI 95%	TER	BLEU	CI 95%	pair-CI 95%	TER
Baseline	14.42	[-0.81, +0.74]	–	75.30	23.60	[-1.03, +0.97]	–	63.56
Para-Sub	14.78	[-0.78, +0.82]	[+0.13, +0.60]	73.75	23.41	[-1.04, +1.00]	[-0.46, +0.09]	63.84
Lattice	**15.44**	[-0.85, +0.84]	[+0.74, +1.30]	**73.06**	**25.20**	[-1.11, +1.15]	[+1.19, +2.01]	**62.37**

Table 2: Comparison between the baseline, "Para-Sub" and our "Lattice" (paraphrase lattice) method.

The paraphrase data set used in our lattice-based and the "Para-Sub" systems is same which is derived from the "Paraphrase Phrase Table"[7] of TER-Plus (Snover et al., 2009). The parameter k in equation 4 is set to 7.

4.2 Paraphrase Filtering

The more edges there are in a lattice, the more complicated the decoding is in the search process. Therefore, in order to reduce the complexity of the lattice and increase decoding speed, we must filter out some potential noise in the paraphrase table. Two measures are taken to optimise the paraphrases when building a paraphrase lattice:

- Firstly, we filter out all the paraphrases whose probability is less than 0.01;
- Secondly, given a source-side input sentence, we retrieve all possible paraphrases and their probabilities for source-side phrases which appear in the paraphrase table. Then we remove the paraphrases which are not occurred in the "phrase table" of the SMT system. This measure intends to avoid adding new "unknown words" to the source-side sentence. After this measure, we can acquire the final paraphrases which can be denoted as a quadruple $< SEN\_ID, Span, Para, Prob >$, where "SEN_ID" indicates the ID of the input sentence, "Span" represents the span of the source-side phrase in the original input sentence, "Para" indicates the paraphrase of the source-side phrase, and "Prob" represents the probability between the source-side phrase and its paraphrase, which is used to set the weight of the edge in the lattice. The quadruple is used to construct the weighted lattice.

4.3 Experimental Results

The experimental results conducted on small and medium-sized data sets are shown in Table 2. The 95% confidence intervals (CI) for BLEU scores are independently computed on each of three systems, while the "pair-CI 95%" are computed relative to the baseline system only for "Para-Sub" and "Lattice" systems. All the significance tests use bootstrap and paired-bootstrap resampling normal approximation methods (Zhang and Vogel, 2004).[8] Improvements are considered to be significant if the left boundary of the confidence interval is larger than zero in terms of the "pair-CI 95%". It can be seen that 1) our "Lattice" system outperforms the baseline by 1.02 and 1.6 absolute (7.07% and 6.78% relative) BLEU points in terms of the 20K and 200K data sets respectively, and our system also decreases the TER scores by 2.24 and 1.19 (2.97% and 1.87% relative) points than the baseline system. In terms of the "pair-CI 95%", the left boundaries for 20K and 200K data are respectively "+0.74" and "+1.19", which indicate that the "Lattice" system is significantly better than the baseline system on these two data sets. 2) The "Para-Sub" system performs slightly better (0.36 absolute BLEU points) than the baseline system on the 20K data set, but slightly worse (0.19 absolute BLEU points) than the baseline on the 200K data set, which indicates that the paraphrase substitution method used in (Callison-Burch et al., 2006) does not work on resource-sufficient data sets. In terms of the "pair-CI 95%", the left boundary for 20K data is "+0.13", which indicates that it is significantly better than the baseline system, while the left boundary is "-0.46" for 200K data, which indicates that the "Para-Sub" system is significantly worse than the baseline system. 3) comparing the "Lattice" system with the "Para-Sub"

[7] `http://www.umiacs.umd.edu/~snover/terp/downloads/terp-pt.v1.tgz`.

[8] `http://projectile.sv.cmu.edu/research/public/tools/bootStrap/tutorial.htm`.

SYS	BLEU	CI 95%	pair-CI 95%	NIST	TER
Baseline	14.04	[-0.73, +0.40]	–	6.50	74.88
Para-Sub	14.13	[-0.56, +0.56]	[-0.18, +0.40]	6.52	74.43
Lattice	**14.55**	[-0.75, +0.32]	[+0.15,+0.83]	**6.55**	**73.28**

Table 3: Comparison between the baseline and our paraphrase lattice method on a large-scale data set.

system, the "pair-CI 95%" for 20K and 200K data are respectively [+0.41, +0.92] and [+1.40, +2.17], which indicates that the "Lattice" system is significantly better than the "Para-Sub" system on these two data sets as well. 4) In terms of the two metrics, our proposed method achieves the best performance, which shows that our method is effective and consistent on different sizes of data.

In order to verify our method on large-scale data, we also perform experiments on 2.1 million sentence-pairs of English-to-Chinese data as described in Section 4.1. The results are shown in Table 3. From Table 3, it can be seen that the "Lattice" system achieves an improvement of 0.51 absolute (3.63% relative) BLEU points and a decrease of 1.6 absolute (2.14% relative) TER points compared to the baseline. In terms of the "pair-CI 95%", the left boundary for the "Lattice" system is "+0.15" which indicates that it is significantly better than the baseline system in terms of BLEU. Interestingly, in our experiment, the "Para-Sub" system also outperforms the baseline on those three automatic metrics. However, in terms of the "pair-CI 95%", the left boundary for the "Para-Sub" system is "-0.18" which indicates that it is not significantly better than the baseline system in terms of BLEU. The results also show that our proposed method is effective and consistent even on a large-scale data set.

It also can be seen that the improvement on 2.1 million sentence-pairs is less than that of the 20K and 200K data sets. That is, as the size of the training data increases, the problems of data sparseness decrease, so that the coverage of the test set by the parallel corpus will correspondingly increase. In this case, the role of paraphrases in decoding becomes a little weaker. On the other hand, it might become a kind of noise to interfere with the exact translation of the original source-side phrases when decoding. Therefore, our proposed method may be more appropriate for language pairs with limited resources.

5 Analysis

5.1 Coverage of Paraphrase Test Set

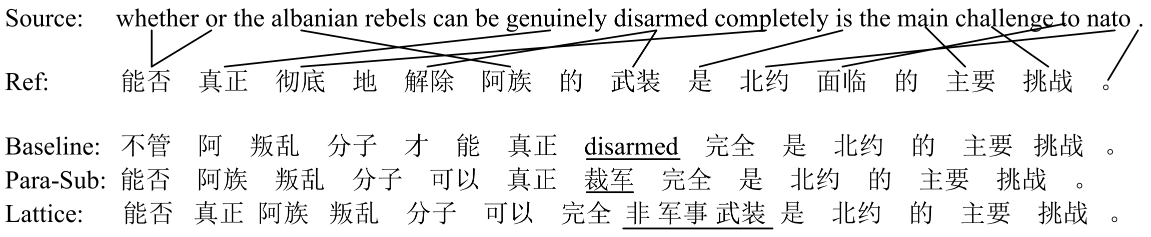
The coverage rate of the test set by the phrase table is an important factor that could influence the translation result, so in this section we examine the characteristics of the updated test set that adds in the paraphrases. We take the 200K data set to examine the coverage issue. Table 4 is an illustration to compare the new coverage and the old coverage (without paraphrases) on medium sized training data.

PL	Tset	PT	New Cov.(%)	Old Cov.(%)
1	9,264	8,994	**97.09**	92.03
2	32,805	25,796	**78.63**	68.40
3	39,918	15,708	**39.35**	34.55
4	42,247	6,479	**15.34**	14.29
5	43,088	2,670	**6.20**	5.99
6	43,066	1,204	**2.80**	2.77
7	42,602	582	1.37	1.36
8	41,865	319	0.76	0.76
9	40,984	233	0.57	0.57
10	40,002	135	0.34	0.34

Table 4: The coverage of the paraphrase-added test set by the phrase table on medium size of the training data.

From Table 4, we can see that the coverage of unigrams, bigrams, trigrams and 4-grams goes up by about 5%, 10%, 5% and 1%, while from 5-grams there is only a slight or no increase in coverage. These results show that 1) most of the paraphrases that are added in are lower-order n-grams; 2) the paraphrases can increase the coverage of the input by handling the unknown words to some extent.

However, we observed that most untranslated words in the "Para-Sub" and "Lattice" systems are still NEs, which shows that in our paraphrase table, there are few paraphrases for the NEs. Therefore, to further improve the translation quality using paraphrases, we also need to acquire the paraphrases for NEs to increase the coverage of unknown words.



Source: whether or the albanian rebels can be genuinely disarmed completely is the main challenge to nato .

Ref: 能否 真正 彻底 地 解除 阿族 的 武装 是 北约 面临 的 主要 挑战 。

Baseline: 不管 阿 叛乱 分子 才 能 真正 disarmed 完全 是 北约 的 主要 挑战 。

Para-Sub: 能否 阿族 叛乱 分子 可以 真正 裁军 完全 是 北约 的 主要 挑战 。

Lattice: 能否 真正 阿族 叛乱 分子 可以 完全 非 军事 武装 是 北约 的 主要 挑战 。

Figure 3: An example from three systems to compare the processing of OOVs

5.2 Analysis on Translation Results

In this section, we give an example to show the effectiveness of using paraphrase lattices to deal with unknown words. The example is evaluated according to both automatic evaluation and human evaluation at sentence level.

See Figure 3 as an illustration of how the paraphrase-based systems process unknown words. According to the word alignments between the source-side sentence and the reference, the word “disarmed” is translated into two Chinese words “解除” and “武装”. These two Chinese words are discontinuous in the reference, so it is difficult for the PB-SMT system to correctly translate the single English word into a discontinuous Chinese phrase. In fact in this example, “disarmed” is an unknown word and it is kept untranslated in the result of the baseline system. In the “Para-Sub” system, it is translated into “裁军” based on a paraphrase pair PP_1 = “disarmed || disarmament || 0.087” and its translation pair T_1 = “disarmament || 裁军”. The number “0.087” is the probability p_1 that indicates to what extent these two words are paraphrases. It can be seen that although “裁军” is quite different from the meaning of “disarmed”, it is understandable for human in some sense. In the “Lattice” system, the word “disarmed” is translated into three Chinese words “非 军事 武装” based on a paraphrase pair PP_2 = “disarmed || demilitarized || 0.099” and its translation pair T_2 = “demilitarized || 非 军事 武装”. The probability p_2 is slightly greater than p_1.

We argue that the reason that the “Lattice” system selects PP_2 and T_2 rather than PP_1 and T_1 is because of the weight estimation in the lattice. That is, PP_2 is more prioritised, while PP_1 is more penalised based on equation (4).

From the viewpoint of human evaluation, the paraphrase pair PP_2 is more appropriate than PP_1, and the translation T_2 is more similar to the original meaning than T_1. The sentence-level automatic evaluation scores for this example in terms of BLEU and TER metrics are shown in Table 5.

SYS	BLEU	TER
Baseline	20.33	66.67
Para-Sub	21.78	**53.33**
Lattice	**23.51**	**53.33**

Table 5: Comparison on sentence-level scores in terms of BLEU and TER metrics.

The BLEU score of the “Lattice” system is much higher than the baseline, and the TER score is quite a bit lower than the baseline. Therefore, from the viewpoint of automatic evaluation, the translation from the “Lattice” system is also better than those from the baseline and “Para-Sub” systems.

6 Conclusions and Future Work

In this paper, we proposed a novel method using paraphrase lattices to facilitate the translation process in SMT. Given an input sentence, our method firstly discovers all possible paraphrases from a paraphrase database for N-grams ($1 <= N <= 10$) in the test set, and then filters out the paraphrases which do not appear in the phrase table in order to avoid adding new unknown words on the input side. We then use the original words and the paraphrases to build a word lattice, and set the weights to prioritise the original edges and penalise the paraphrase edges. Finally, we import the lattice into the decoder to perform lattice decoding. The experiments are conducted on English-to-Chinese translation using the FBIS data set with small and medium-sized amounts of data, and on a large-scale corpus of 2.1

million sentence pairs. We also performed comparative experiments for the baseline, the "Para-Sub" system and our paraphrase lattice-based system. The experimental results show that our proposed system significantly outperforms the baseline and the "Para-Sub" system, and the effectiveness is consistent on the small, medium and large-scale data sets.

As for future work, firstly we plan to propose a pruning algorithm for the duplicate paths in the lattice, which will track the edge generation with respect to the path span, and thus eliminate duplicate paths. Secondly, we plan to experiment with another feature function in the log-linear model to discount words derived from paraphrases, and use MERT to assign an appropriate weight to this feature function.

Acknowledgments

Many thanks to the reviewers for their insightful comments and suggestions. This work is supported by Science Foundation Ireland (Grant No. 07/CE/I1142) as part of the Centre for Next Generation Localisation (www.cngl.ie) at Dublin City University.

References

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *43rd Annual meeting of the Association for Computational Linguistics*, Ann Arbor, MI, pages 597–604.

Chris Callison-Burch, Philipp Koehn and Miles Osborne. 2006. Improved statistical machine translation using paraphrases. In *Proceedings of HLT-NAACL 2006: Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL*, NY, USA, pages 17–24.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *43rd Annual meeting of the Association for Computational Linguistics*, Ann Arbor, MI, pages 263–270.

Philipp Koehn, Franz Josef Och and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL 2003: conference combining Human Language Technology conference series and the North American Chapter of the Association for Computational Linguistics conference series*, Edmonton, Canada, pages 48–54.

Philipp Koehn, Hieu Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, Wade Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *ACL 2007: demo and poster sessions*, Prague, Czech Republic, pages 177–180.

Preslav Nakov. 2008. Improving English-Spanish statistical machine translation: experiments in domain adaptation, sentence paraphrasing, tokenization, and recasing. In *Proceedings of ACL-08:HLT. Third Workshop on Statistical Machine Translation*, Columbus, Ohio, USA, pages 147–150.

Franz Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *41st Annual meeting of the Association for Computational Linguistics*, Sapporo, Japan, pages 160–167.

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

Kishore Papineni, Salim Roukos, Todd Ward and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *40th Annual meeting of the Association for Computational Linguistics*, Philadelphia, PA, pages 311–318.

Josh Schroeder, Trevor Cohn and Philipp Koehn. 2009. Word Lattices for Multi-source Translation. In *Proceedings of the 12th Conference of the European Chapter of the ACL*, Athens, Greece, pages 719–727.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*, Cambridge, pages 223–231.

Matthew Snover, Nitin Madnani, Bonnie J.Dorr and Richard Schwartz. 2009. Fluency, adequacy, or HTER? Exploring different human judgments with a tunable MT metric. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, Athens, Greece, pages 259–268.

Ying Zhang and Stephan Vogel. 2004. Measuring Confidence Intervals for the Machine Translation Evaluation Metrics. In *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*, pages 85–94.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of HLT-NAACL 2006: Proceedings of the Workshop on Statistical Machine Translation*, New York, pages 138–141.

Mining Name Translations from Entity Graph Mapping[*]

Gae-won You[†] Seung-won Hwang[†] Young-In Song[‡] Long Jiang[‡] Zaiqing Nie[‡]

[†]Pohang University of Science and Technology, Pohang, Republic of Korea
{gwyou,swhwang}@postech.ac.kr

[‡]Microsoft Research Asia, Beijing, China
{yosong,longj,znie}@microsoft.com

Abstract

This paper studies the problem of mining entity translation, specifically, mining English and Chinese name pairs. Existing efforts can be categorized into (a) a transliteration-based approach leveraging phonetic similarity and (b) a corpus-based approach exploiting bilingual co-occurrences, each of which suffers from inaccuracy and scarcity respectively. In clear contrast, we use unleveraged resources of monolingual entity co-occurrences, crawled from entity search engines, represented as two entity-relationship graphs extracted from two language corpora respectively. Our problem is then abstracted as finding correct mappings across two graphs. To achieve this goal, we propose a holistic approach, of exploiting both transliteration similarity and monolingual co-occurrences. This approach, building upon monolingual corpora, complements existing corpus-based work, requiring scarce resources of parallel or comparable corpus, while significantly boosting the accuracy of transliteration-based work. We validate our proposed system using real-life datasets.

1 Introduction

Entity translation aims at mapping the entity names (*e.g.*, people, locations, and organizations) in source language into their corresponding names in target language. While high quality entity translation is essential in cross-lingual information access and translation, it is non-trivial to achieve, due to the challenge that entity translation, though typically bearing pronunciation similarity, can also be arbitrary, *e.g.*, Jackie Chan and 成龙 (pronounced Cheng Long). Existing efforts to address these challenges can be categorized into transliteration- and corpus-based approaches. Transliteration-based approaches (Wan and Verspoor, 1998; Knight and Graehl, 1998) identify translations based on pronunciation similarity, while corpus-based approaches mine bilingual co-occurrences of translation pairs obtained from parallel (Kupiec, 1993; Feng et al., 2004) or comparable (Fung and Yee, 1998) corpora, or alternatively mined from bilingual sentences (Lin et al., 2008; Jiang et al., 2009). These two approaches have complementary strength– transliteration-based similarity can be computed for any name pair but cannot mine translations of little (or none) phonetic similarity. Corpus-based similarity can support arbitrary translations, but require highly scarce resources of bilingual co-occurrences, obtained from parallel or comparable bilingual corpora.

In this paper, we propose a holistic approach, leveraging both transliteration- and corpus-based similarity. Our key contribution is to replace the use of scarce resources of bilingual co-occurrences with the use of untapped and significantly larger resources of monolingual co-occurrences for translation. In particular, we extract monolingual co-occurrences of entities from English and Chinese Web corpora, which are readily available from entity search engines such as PeopleEntityCube[1], deployed by Microsoft Research Asia. Such engine

[*] This work was done when the first two authors visited Microsoft Research Asia.

[1] http://people.entitycube.com

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 430–439,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

automatically extracts people names from text and their co-occurrences to retrieve related entities based on co-occurrences. To illustrate, Figure 1(a) demonstrates the query result for “Bill Gates,” retrieving and visualizing the “entity-relationship graph” of related people names that frequently co-occur with Bill in English corpus. Similarly, entity-relationship graphs can be built over other language corpora, as Figure 1(b) demonstrates the corresponding results for the same query, from Renlifang[2] on Chinese Web corpus. From this point on, for the sake of simplicity, we refer to English and Chinese graphs, simply as G_e and G_c respectively. Though we illustrate with English-Chinese pairs in the paper, our method can be easily adapted to other language pairs.

In particular, we propose a novel approach of abstracting entity translation as a graph matching problem of two graphs G_e and G_c in Figures 1(a) and (b). Specifically, the similarity between two nodes v_e and v_c in G_e and G_c is initialized as their transliteration similarity, which is iteratively refined based on relational similarity obtained from monolingual co-occurrences. To illustrate this, an English news article mentioning “Bill Gates” and “Melinda Gates” evidences a relationship between the two entities, which can be quantified from their co-occurrences in the entire English Web corpus. Similarly, we can mine Chinese news articles to obtain the relationships between “比尔·盖茨” and “梅琳达·盖茨”. Once these two bilingual graphs of people and their relationships are harvested, entity translation can leverage these parallel relationships to further evidence the mapping between translation pairs, as Figure 1(c) illustrates.

To highlight the advantage of our proposed approach, we compare our results with commercial machine translators (1) Engkoo[3] developed in Microsoft Research Asia and (2) Google Translator[4]. In particular, Figure 2 reports the precision for two groups– “heads” that belong to top-100 popular people (determined by the number of hits), among randomly sampled 304 people names[5] from six graph pairs of size 1,000 each, and the remaining “tails”. Commercial translators such as Google, leveraging bilingual co-occurrences that are scarce for tails, show significantly lower precision for tails. Meanwhile, our work, depending solely on monolingual co-occurrences, shows high precision, for both heads and tails.

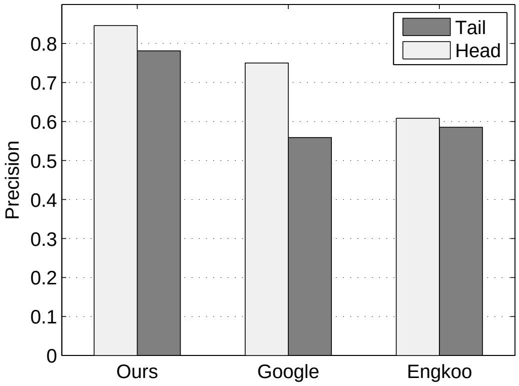


Figure 2: Comparison for Head and Tail datasets

Our focus is to boost translation accuracy for long tails with non-trivial Web occurrences in each monolingual corpus, but not with much bilingual co-occurrences, *e.g.*, researchers publishing actively in two languages but not famous enough to be featured in multi-lingual Wikipedia entries or news articles. As existing translators are already highly accurate for popular heads, this focus well addresses the remaining challenges for entity translation.

To summarize, we believe that this paper has the following contributions:

- We abstract entity translation problem as a graph mapping between entity-relationship graphs in two languages.
- We develop an effective matching algorithm leveraging both pronunciation and co-occurrence similarity. This holistic approach complements existing approaches and enhances the translation coverage and accuracy.
- We validate the effectiveness of our approach using various real-life datasets.

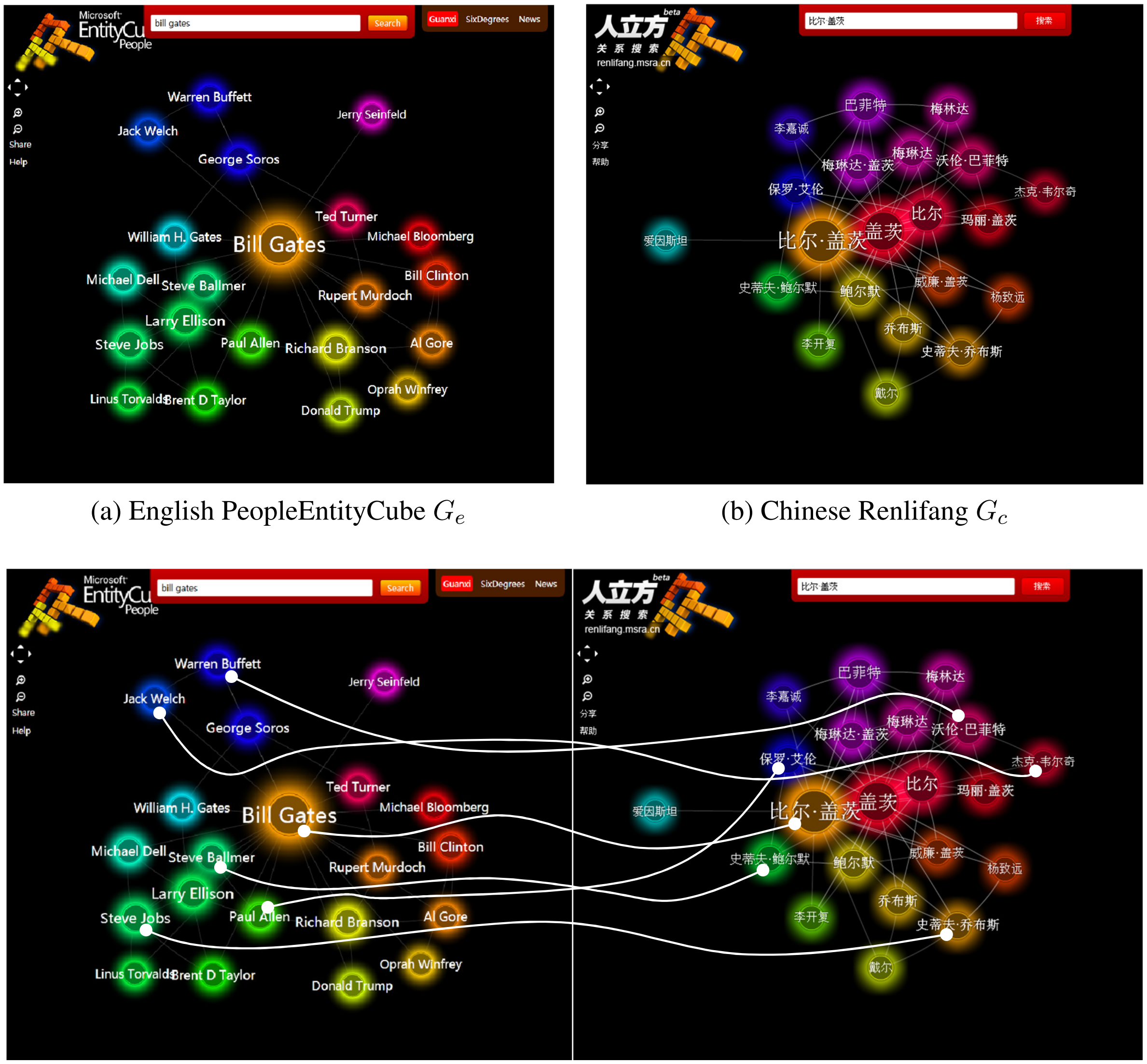
The rest of this paper is organized as follows. Section 2 reviews existing work. Section 3 then develops our framework. Section 4 reports experimental results and Section 5 concludes our work.

[2] http://renlifang.msra.cn

[3] http://www.engkoo.com

[4] http://translate.google.com

[5] See Section 4 for the sampling process.



(a) English PeopleEntityCube G_e

(b) Chinese Renlifang G_c

(c) Abstracting translation as graph mapping

Figure 1: Illustration of entity-relationship graphs

2 Related Work

In this section, we first survey related efforts, categorized into transliteration-based and corpus-based approaches. Our approach leveraging both is complementary to these efforts.

2.1 Transliteration-based Approaches

Many name translations are loosely based on phonetic similarity, which naturally inspires transliteration-based translation of finding the translation with the closest pronunciation similarity, using either rule-based (Wan and Verspoor, 1998) or statistical (Knight and Graehl, 1998; Li et al., 2004) approaches. However, people are free to designate arbitrary bilingual names of little (or none) phonetic similarity, for which the transliteration-based approach is not effective.

2.2 Corpus-based Approaches

Corpus-based approach can mine arbitrary translation pairs, by mining bilingual co-occurrences from parallel and comparable bilingual corpora. Using parallel corpora (Kupiec, 1993; Feng et al., 2004), *e.g.*, bilingual Wikipedia entries on the same person, renders high accuracy but suffers from high scarcity. To alleviate such scarcity, (Fung and Yee, 1998; Shao and Ng, 2004) explore a more vast resource of comparable corpora, which share no parallel document- or sentence-alignments as in parallel corpora but describe similar contents in two languages, *e.g.*, news articles on the same event. Alternatively, (Lin et al., 2008) extracts bilingual co-occurrences from bilingual sentences, such as annotating terms with their corresponding translations in English inside parentheses. Similarly, (Jiang et al., 2009) identifies potential translation pairs from bilingual sentences using lexical pattern analysis.

2.3 Holistic Approaches

The complementary strength of the above two approaches naturally calls for a holistic approach, such as recent work combining transliteration- and corpus-based similarity mining bilingual co-occurrences using general search engines. Specifically, (Al-Onaizan and Knight, 2002) uses transliteration to generate candidates and then web corpora to identify translations. Later, (Jiang et al., 2007) enhances to use transliteration to guide web mining.

Our work is also a holistic approach, but leveraging significantly larger corpora, specifically by exploiting monolingual co-occurrences. Such expansion enables to translate "long-tail" people entities with non-trivial Web occurrences in each monolingual corpus, but not much bilingual co-occurrences. Specifically, we initialize name pair similarity using transliteration-based approach, and iteratively reinforces base similarity using relational similarity.

3 Our Framework

Given two graphs $G_e = (V_e, E_e)$ and $G_c = (V_c, E_c)$ harvested from English and Chinese corpora respectively, our goal is to find translation pairs, or a set S of matching node pairs such that $S \subseteq V_e \times V_c$. Let R be a $|V_e|$-by-$|V_c|$ matrix where each R_{ij} denotes the similarity between two nodes $i \in V_e$ and $j \in V_c$.

Overall, with the matrix R, our approach consists of the following three steps, as we will discuss in the following three sections respectively:

1. **Initialization:** computing base translation similarities R_{ij} between two entity nodes using transliteration similarity

2. **Reinforcement model:** reinforcing the translation similarities R_{ij} by exploiting the monolingual co-occurrences

3. **Matching extraction:** extracting the matching pairs from the final translation similarities R_{ij}

3.1 Initialization with Transliteration

We initialize the translation similarity R_{ij} as the transliteration similarity. This section explains how to get the transliteration similarity between English and Chinese names using an unsupervised approach.

Formally, let an English name $N_e = (e_1, e_2, \cdots, e_n)$ and a Chinese name $N_c = (c_1, c_2, \cdots, c_m)$ be given, where e_i is an English word and N_e is a sequence of the words, and c_i is a Chinese character and N_c is a sequence of the characters. Our goal is to compute a score indicating the similarity between the pronunciations of the two names.

We first convert N_c into its Pinyin representation $PY_c = (s_1, s_2, \cdots, s_m)$, where s_i is the Pinyin representation of c_i. Pinyin is the romanization representation of pronunciation of Chinese character. For example, the Pinyin representation of $N_e =$ ("Barack", "Obama") is $PY_c =$("ba", "la", "ke", "ao", "ba", "ma"). The Pinyin representations of Chinese characters can be easily obtained from Chinese character pronunciation dictionary. In our experiments, we use an in-house dictionary, which contains pronunciations of $20,774$ Chinese characters. For the Chinese characters having multiple pronunciations, we only use the most popular one.

Calculation of transliteration similarity between N_e and N_c is now transformed to calculation of pronunciation similarity between N_e and PY_c. Because letters in Chinese Pinyins and English strings are pronounced similarly, we can further approximate pronunciation similarity between N_e and PY_c using their spelling similarity. In this paper, we use Edit Distance (ED) to measure the spelling similarity. Moreover, since words in N_e are transliterated into characters in PY_c independently, it is more accurate to compute the ED between N_e and PY_c, *i.e.*, $ED_{name}(N_e, PY_c)$, as the sum of the EDs of all component transliteration pairs, *i.e.*, every e_i in N_e and its corresponding transliteration (s_i) in PY_c. In other words, we need to first align all s_j's in PY_c with corresponding e_i in N_e based on whether they are translations of each other. Then based on the alignment, we can calculate $ED_{name}(N_e, PY_c)$ using the following formula.

$$ED_{name}(N_e, PY_c) = \sum_i ED(e_i, es_i) \quad (1)$$

where es_i is a string generated by concatenating all s_i's that are aligned to e_i and $ED(e_i, es_i)$ is the Edit Distance between e_i and es_i, *i.e.*, the minimum number of edit operations (including insertion, deletion and substitution) needed to transform e_i into es_i. Because an English word usually consists of multiple syllables but every Chinese character consists of only one syllable, when aligning e_i's with s_j's, we add the constraint that each e_i is allowed to be aligned with 0 to 4 s_i's but each s_i can only be aligned with 0 to 1 e_i. To get the alignment between PY_c and N_e which has the minimal $ED_{name}(N_e, PY_c)$, we use a Dynamic Programming based algorithm as defined in the following formula:

$$\begin{aligned}
&ED_{name}(N_e^{1,i}, PY_c^{1,j}) = \min(\\
&\quad ED_{name}(N_e^{1,i-1}, PY_c^{1,j}) + Len(e_i), \\
&\quad ED_{name}(N_e^{1,i}, PY_c^{1,j-1}) + Len(s_j), \\
&\quad ED_{name}(N_e^{1,i-1}, PY_c^{1,j-1}) + ED(e_i, s_j), \\
&\quad ED_{name}(N_e^{1,i-1}, PY_c^{1,j-2}) + ED(e_i, PY_c^{j-1,j}), \\
&\quad ED_{name}(N_e^{1,i-1}, PY_c^{1,j-3}) + ED(e_i, PY_c^{j-2,j}), \\
&\quad ED_{name}(N_e^{1,i-1}, PY_c^{1,j-4}) + ED(e_i, PY_c^{j-3,j}))
\end{aligned}$$

where, given a sequence $X = (x_1, x_2, \cdots)$ such that x_i is a word, $X^{i,j}$ is the subsequence $(x_i, x_{i+1}, \cdots, x_j)$ of X and $Len(X)$ is the number of letters except spaces in the sequence X. For instance, the minimal Edit Distance between the English name "Barack Obama" and the Chinese Pinyin representation "ba la ke ao ba ma" is 4, as the best alignment is: "Barack" ↔ "ba la ke" (ED: 3), "Obama" ↔ "ao ba ma" (ED: 1). Finally the transliteration similarity between N_c and N_e is calculated using the following formula.

$$Sim_{tl}(N_c, N_e) = 1 - \frac{ED_{name}(N_e, PY_c)}{Len(PY_c) + Len(N_e)} \quad (2)$$

For example, Sim_{tl}("Barack Obama", "巴拉克·奥巴马") is $1 - \frac{4}{11+12} = 0.826$.

3.2 Reinforcement Model

From the initial similarity, we model our problem as an iterative approach that iteratively reinforces the similarity R_{ij} of the nodes i and j from the matching similarities of their neighbor nodes u and v.

The basic intuition is built on exploiting the similarity between monolingual co-occurrences of two different languages. In particular, we assume two entities with strong relationship co-occur frequently in both corpora. In order to express this intuition, we formally define an iterative reinforcement model as follows. Let R_{ij}^t denote the similarity of nodes i and j at t-th iteration:

$$R_{ij}^{t+1} = \lambda \sum_{(u,v)_k \in B^t(i,j,\theta)} \frac{R_{uv}^t}{2^k} + (1-\lambda) R_{ij}^0 \quad (3)$$

The model is expressed as a linear combination of (a) the relational similarity $\sum R_{uv}^t / 2^k$ and (b) transliteration similarity R_{ij}^0. (λ is the coefficient for interpolating two similarities.)

In the relational similarity, $B^t(i,j,\theta)$ is an ordered set of the best matching pairs between neighbor nodes of i and ones of j such that $\forall (u,v)_k \in B^t(i,j,\theta), R_{uv}^t \geq \theta$, where $(u,v)_k$ is the matching pair with k-th highest similarity score. We consider (u,v) with similarity over some threshold θ, or $R_{uv}^t \geq \theta$, as a matching pair. In this neighbor matching process, if many-to-many matches exist, we select only one with the greatest matching score. Figure 3 describes such matching process more formally. $N(i)$ and $N(j)$ are the sets of neighbor nodes of i and j, respectively, and H is a priority queue sorting pairs in the decreasing order of similarity scores.

Meanwhile, note that, in order to express that the confidence for matching (i,j) progressively converges as the number of matched neighbors increases, we empirically use decaying coefficient $1/2^k$ for R_{uv}^t, because $\sum_{k=1}^{\infty} 1/2^k = 1$.

3.3 Matching Extraction

After the convergence of the above model, we get the $|V_e|$-by-$|V_c|$ similarity matrix R^{∞}. From this matrix, we extract one-to-one matches maximizing the overall similarity.

More formally, this problem can be stated as the *maximum weighted bipartite matching* (West,

```
$B^t(i,j,\theta) \leftarrow \{\}$
$\forall u \in N(i), \forall v \in N(j) : H.push(u,v;R^t_{uv})$
while $H$ is not empty do
    $(u,v;s) \leftarrow H.pop()$
    if $s < \theta$ then
        break
    end if
    if neither $u$ nor $v$ are matched yet then
        $B^t(i,j,\theta) \leftarrow B^t(i,j,\theta) \cup \{(u,v)\}$
    end if
end while
return $B^t(i,j,\theta)$
```

Figure 3: How to get the ordered set $B^t(i,j,\theta)$

2000)– Given two groups of entities V_e and V_c from the two graphs G_e and G_c, we can build a *weighted bipartite graph* is $G = (V,E)$, where $V = V_e \cup V_c$ and E is a set of edges (u,v) with weight R^∞_{uv}. To filter out null alignment, we construct only the edges with weight $R^\infty_{uv} \geq \delta$. From this bipartite graph, the maximum weighted bipartite matching problem finds a set of pairwise non-adjacent edges $S \subseteq E$ such that $\sum_{(u,v)\in S} R^\infty_{uv}$ is the maximum. Well-known algorithms include Hungarian algorithm with time complexity of $O(|V|^2 \log |V| + |V||E|)$ (West, 2000).

In this paper, to speed up processing, we consider a greedy alternative with the following steps– (1) choose the pair with the highest similarity score, (2) remove the corresponding row and column from the matrix, and (3) repeat (1) and (2) until their matching scores are over a specific threshold δ.

4 Experiments

This section reports our experimental results to evaluate our proposed approach. First, we report our experimental setting in Section 4.1. Second, we validate the effectiveness and the scalability of our approach over a real-life dataset in Section 4.2.

4.1 Experimental Settings

This section describes (1) how we collect the English and Chinese EntityCube datasets, (2) how to build ground-truth test datasets for evaluating our framework, and (3) how to set up three parameters λ, θ, and δ.

First, we crawled $G_e = (V_e, E_e)$ and $G_c = (V_c, E_c)$ from English and Chinese EntityCubes. Specifically, we built a graph pairs (G_e, G_c) expanding from a "seed pair" of nodes $s_e \in V_e$ and $s_c \in V_c$ until the number of nodes for each graph becomes 1,000[6]. More specifically, when we build a graph G_e by expanding from s_e, we use a queue Q. We first initialize Q by pushing the seed node s_e. We then iteratively pop a node v_e from Q, save v_e into V_e, and push its neighbor nodes in decreasing order of co-occurrence scores with v_e. Similarly, we can get G_c from a counterpart seed node v_c. By using this procedure, we built six graph pairs from six different seed pairs. In particular, the six seed nodes are English names and its corresponding Chinese names representing a wide range of occupation domains (*e.g.*, 'Barack Obama,' 'Bill Gates,' 'Britney Spears,' 'Bruno Senna,' 'Chris Paul,' and 'Eminem') as Table 1 depicts. Meanwhile, though we demonstrate the effectiveness of the proposed method for mining name translations in Chinese and English languages, the method can be easily adapted to other language pairs.

Table 1: Summary for graphs and test datasets obtained from each seed pair

i	$\lvert V_e \rvert, \lvert V_c \rvert$	$\lvert T_i \rvert$	English Name	Chinese Name
1	1,000	51	Barack Obama	巴拉克·奥巴马
2	1,000	52	Bill Gates	比尔·盖茨
3	1,000	40	Britney Spears	布兰妮·斯皮尔斯
4	1,000	53	Bruno Senna	布鲁诺·塞纳
5	1,000	51	Chris Paul	克里斯·保罗
6	1,000	57	Eminem	艾米纳姆

Second, we manually searched for about 50 "ground-truth" matched translations for each graph pair to build test datasets T_i, by randomly selecting nodes within two hops[7] from the seed pair (s_e, s_c), since nodes outside two hops may include nodes whose neighbors are not fully crawled. More specifically, due to our crawling process expanding to add neighbors from the seed, the nodes close to the seed have all the neighbors they would have in the full graph, while those far from the node may not. In order to pick the nodes that well represent the actual

[6] Note, this is just a default setting, which we later increase for scalability evaluation in Figure 6.

[7] Note that the numbers of nodes within two hops in G_e and G_c are 327 and 399 on average respectively.

neighbors, we built test datasets among those within two hops. However, this crawling is used for the evaluation sake only, and thus does not suggest the bias in our proposed framework. Table 1 describes the size of such test dataset for each graph pair.

Lastly, we set up the three parameters λ, θ, and δ using 6-fold cross validation with 6 test datasets T_i's of the graphs. More specifically, for each dataset T_i, we decide λ_i and θ_i such that average MRR for the other 5 test datasets is maximized. (About MRR, see more details of Equation (4) in Section 4.2.) We then decide δ_i such that average F1-score is maximized. Figure 4 shows the average MRR for λ_i and θ_i with default values $\theta = 0.66$ and $\lambda = 0.2$. Based on these results, we set λ_i with values $\{0.2, 0.15, 0.2, 0.15, 0.2, 0.15\}$ that optimize MRR in datasets $T_1, \ldots T_6$, and similarly θ_i with $\{0.67, 0.65, 0.67, 0.67, 0.65, 0.67\}$. We also set δ_i with values $\{0.63, 0.63, 0.61, 0.61, 0.61, 0.61\}$ optimizing F1-score with the same default values $\lambda = 0.2$ and $\theta = 0.66$. We can observe the variances of optimal parameter setting values are low, which suggests the robustness of our framework.

4.2 Experimental Results

This section reports our experimental results using the evaluation datasets explained in previous section. For each graph pair, we evaluated the effectiveness of (1) reinforcement model using MRR measure in Section 4.2.1 and (2) overall framework using precision, recall, and F1 measures in Section 4.2.2. We also validated (3) scalability of our framework over larger scale of graphs (with up to five thousand nodes) in Section 4.2.3. (In all experimental results, **Bold numbers** indicate the best performance for each metric.)

4.2.1 Effectiveness of reinforcement model

We evaluated the reinforcement model over MRR (Voorhees, 2001), the average of the reciprocal ranks of the query results as the following formula:

$$\text{MRR} = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{rank_q} \tag{4}$$

Each q is a ground-truth matched pair (u, v) such that $u \in V_e$ and $v \in V_c$, and $rank_q$ is the rank of the similarity score of R_{uv} among all R_{uk}'s such that $k \in V_c$. Q is a set of such queries. By comparing MRRs for two matrices R^0 and R^∞, we can validate the effectiveness of the reinforcement model.

- Baseline matrix (R^0): using only the transliteration similarity score, *i.e.*, without reinforcement
- Reinforced matrix (R^∞): using the reinforced similarity score obtained from Equation (3)

Table 2: MRR of baseline and reinforced matrices

Set	MRR	
	Baseline R^0	Reinforced R^∞
T_1	0.6964	**0.8377**
T_2	0.6213	**0.7581**
T_3	0.7095	**0.7989**
T_4	0.8159	**0.8378**
T_5	0.6984	**0.8158**
T_6	0.5982	**0.8011**
Average	0.6900	**0.8082**

We empirically observed that the iterative model converges within 5 iterations. In all experiments, we used 5 iterations for the reinforcement.

Table 2 summarizes our experimental results. As these figures show, MRR scores significantly increase after applying our reinforcement model except for the set T_4 (on average from 69% to 81%), which indirectly shows the effectiveness of our reinforcement model.

4.2.2 Effectiveness of overall framework

Based on the reinforced matrix, we evaluated the effectiveness of our overall matching framework using the following three measures–(1) **precision**: how accurately the method returns matching pairs, (2) **recall**: how many the method returns correct matching pairs, and (3) **F1-score**: the harmonic mean of precision and recall. We compared our approach with a baseline, mapping two graphs with only transliteration similarity.

- Baseline: in matching extraction, using R^0 as the similarity matrix by bypassing the reinforcement step
- Ours: using R^∞, the similarity matrix converged by Equation (3)

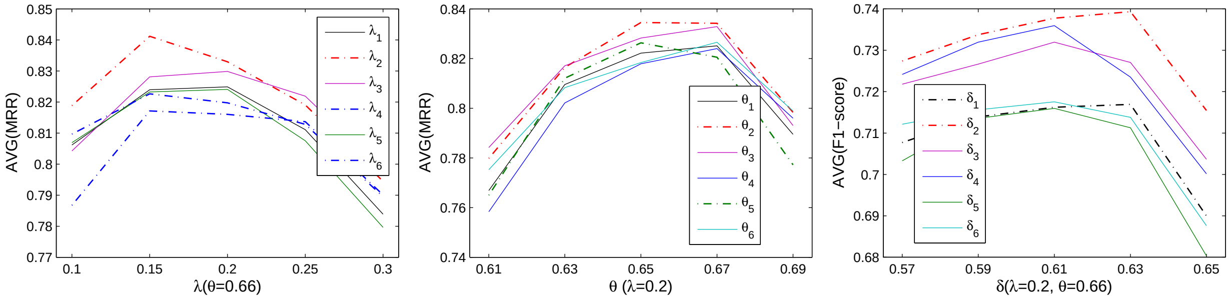


Figure 4: Parameter setup for λ, θ, and δ

In addition, we compared ours with the machine translators of Engkoo and Google. Table 3 summarizes our experimental results.

As this table shows, our approach results in the highest precision (about 80% on average) without compromising the best recall of Google, *i.e.*, 61% of Google vs. 63% of ours. Overall, our approach outperforms others in all three measures.

Meanwhile, in order to validate the translation accuracy over popular head and long-tail, as discussed in Section 1, we separated the test data into two groups and analyzed the effectiveness separately. Figure 5 plots the number of hits returned for the names from Google search engine. According to the distribution, we separate the test data into top-100 popular people with the highest hits and the remaining, denoted *head* and *tail*, respectively.

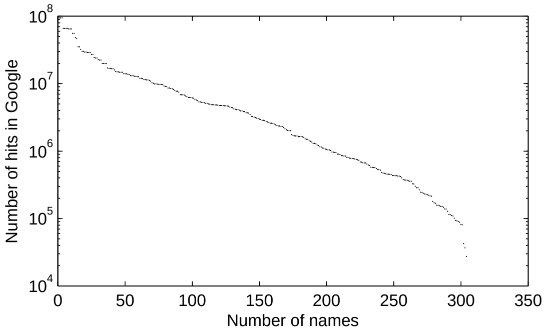


Figure 5: Distribution over number of hits

Table 4 shows the effectiveness with both datasets, respectively. As difference of the effectiveness between tail and head (denoted *diff*) with respect to three measures shows, our approach shows stably high precision, for both heads and tails.

4.2.3 Scalability

To validate the scalability of our approach, we evaluated the effectiveness of our approach over the number of nodes in two graphs. We built larger six graph pairs (G_e, G_c) by expanding them from the seed pairs further until the number of nodes reaches 5,000. Figure 6 shows the number of matched translations according to such increase. Overall, the number of matched pairs linearly increases as the number of nodes increases, which suggests scalability. The ratio of node overlap in two graphs is about between 7% and 9% of total node size.

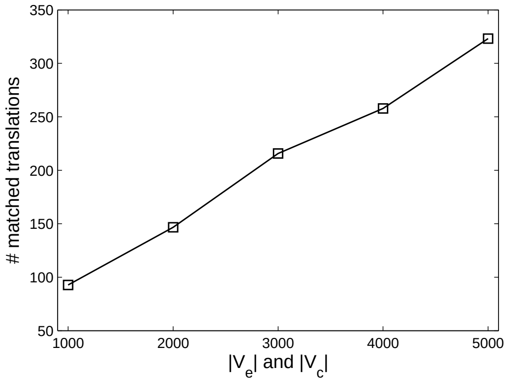


Figure 6: Matched translations over $|V_e|$ and $|V_c|$

5 Conclusion

This paper abstracted name translation problem as a matching problem of two entity-relationship graphs. This novel approach complements existing name translation work, by not requiring rare resources of parallel or comparable corpus yet outperforming the state-of-the-art. More specifically, we combine bilingual phonetic similarity and monolingual Web co-occurrence similarity, to compute a holistic notion of entity similarity. To achieve this goal, we de-

Table 3: Precision, Recall, and F1-score of Baseline, Engkoo, Google, and Ours over test sets T_i

Set	Precision				Recall				F1-score			
	Engkoo	Google	Baseline	Ours	Engkoo	Google	Baseline	Ours	Engkoo	Google	Baseline	Ours
T_1	0.5263	0.4510	0.5263	**0.8974**	0.3922	0.4510	0.1961	**0.6863**	0.4494	0.4510	0.2857	**0.7778**
T_2	0.7551	0.75	0.7143	**0.8056**	0.7115	**0.75**	0.2885	0.5577	0.7327	**0.75**	0.4110	0.6591
T_3	0.5833	0.7925	0.5556	**0.7949**	0.5283	**0.7925**	0.1887	0.5849	0.5545	**0.7925**	0.2817	0.6739
T_4	0.5	0.45	**0.7368**	0.7353	0.425	0.45	0.35	**0.625**	0.4595	0.45	0.4746	**0.6757**
T_5	0.6111	0.3137	0.5	**0.7234**	0.4314	0.3137	0.1765	**0.6667**	0.5057	0.3137	0.2609	**0.6939**
T_6	0.5636	**0.8947**	0.6	0.8605	0.5438	**0.8947**	0.1053	0.6491	0.5536	**0.8947**	0.1791	0.74
AVG	0.5899	0.6086	0.6055	**0.8028**	0.5054	0.6086	0.2175	**0.6283**	0.5426	0.6086	0.3155	**0.7034**

Table 4: Precision, Recall, and F1-score of Engkoo, Google, and Ours with head and tail datasets

Method	Precision			Recall			F1-score		
	head	tail	diff	head	tail	diff	head	tail	diff
Engkoo	0.6082	0.5854	**0.0229**	0.59	0.4706	0.1194	0.5990	0.5217	0.0772
Google	0.75	0.5588	0.1912	**0.75**	0.5588	0.1912	**0.75**	0.5588	0.1912
Ours	**0.8462**	**0.7812**	0.0649	0.66	**0.6127**	**0.0473**	0.7416	**0.6868**	**0.0548**

veloped a graph alignment algorithm that iteratively reinforces the matching similarity exploiting relational similarity and then extracts correct matches. Our evaluation results empirically validated the accuracy of our algorithm over real-life datasets, and showed the effectiveness on our proposed perspective.

Acknowledgments

This work was supported by Microsoft Research Asia NLP theme funding and MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute for Information Technology Advancement) (IITA-2009-C1090-0902-0045).

References

Yaser Al-Onaizan and Kevin Knight. 2002. Translating Named Entities Using Monolingual and Bilingual Resources. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL'02)*, pages 400–408. Association for Computational Linguistics.

Donghui Feng, Yajuan Lü, and Ming Zhou. 2004. A New Approach for English-Chinese Named Entity Alignment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'04)*, pages 372–379. Association for Computational Linguistics.

Pascale Fung and Lo Yuen Yee. 1998. An IR Approach for Translating New Words from Nonparallel,Comparable Texts. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING'98)*, pages 414–420. Association for Computational Linguistics.

Long Jiang, Ming Zhou, Lee feng Chien, and Cheng Niu. 2007. Named Entity Translation with Web Mining and Transliteration. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 1629–1634. Morgan Kaufmann Publishers Inc.

Long Jiang, Shiquan Yang, Ming Zhou, Xiaohua Liu, and Qingsheng Zhu. 2009. Mining Bilingual Data from the Web with Adaptively Learnt Patterns. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL'09)*, pages 870–878. Association for Computational Linguistics.

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

Julian Kupiec. 1993. An Algorithm for finding Noun Phrase Correspondences in Bilingual Corpora. In *Proceedings of the 31th Annual Meeting of the Association for Computational Linguistics (ACL'93)*, pages 17–22. Association for Computational Linguistics.

Haizhou Li, 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 (ACL'04)*, pages 159–166. Association for Computational Linguistics.

Dekang Lin, Shaojun Zhao, Benjamin Van Durme, and Marius Pasca. 2008. Mining Parenthetical Translations from the Web by Word Alignment. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL'08)*, pages 994–1002. Association for Computational Linguistics.

Li Shao and Hwee Tou Ng. 2004. Mining New Word Translations from Comparable Corpora. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING'04)*, pages 618–624. Association for Computational Linguistics.

Ellen M. Voorhees. 2001. The trec question answering track. *Natural Language Engineering*, 7(4):361–378.

Stephen Wan and Cornelia Maria Verspoor. 1998. Automatic English-Chinese Name Transliteration for Development of Multilingual Resources. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING'98)*, pages 1352–1356. Association for Computational Linguistics.

Douglas Brent West. 2000. *Introduction to Graph Theory*. Prentice Hall, second edition.

Non-isomorphic Forest Pair Translation

Hui Zhang[1, 2, 3] **Min Zhang**[1] **Haizhou Li**[1] **Eng Siong Chng**[2]

[1]Institute for Infocomm Research
[2]Nanyang Technological University
[3]USC Information Science Institute
huizhang.fuan@gmail.com {mzhang, hli}@i2r.a-star.edu.sg aseschng@ntu.edu.sg

Abstract

This paper studies two issues, non-isomorphic structure translation and target syntactic structure usage, for statistical machine translation in the context of forest-based tree to tree sequence translation. For the first issue, we propose a novel non-isomorphic translation framework to capture more non-isomorphic structure mappings than traditional tree-based and tree-sequence-based translation methods. For the second issue, we propose a parallel space searching method to generate hypothesis using tree-to-string model and evaluate its syntactic goodness using tree-to-tree/tree sequence model. This not only reduces the search complexity by merging spurious-ambiguity translation paths and solves the data sparseness issue in training, but also serves as a syntax-based target language model for better grammatical generation. Experiment results on the benchmark data show our proposed two solutions are very effective, achieving significant performance improvement over baselines when applying to different translation models.

1 Introduction

Recently syntax-based methods have achieved very promising results and attracted increasing interests in statistical machine translation (SMT) research community due to their ability to provide informative context structure information and convenience in carrying out word transformation and sub-span reordering. Fundamentally, syntax-based SMT views translation as a structural transformation process. Generally speaking, from modeling viewpoint, a syntax-based model tries to convert the source structures into target structures iteratively and recursively while from decoding viewpoint a syntax-based system segments an input tree/forest into many sub-fragments, translates each of them separately, combines the translated sub-fragments and then finds out the best combinations. Therefore, from bilingual viewpoint, we face two fundamental problems: the mapping between bilingual structures and the way of carrying out the target structures combination.

For the first issue, a number of models have been proposed to model the structure mapping between tree and string (Galley *et al.*, 2004; Liu *et al.*, 2006; Yamada and Knight, 2001; DeNeefe and Knight, 2009) and between tree and tree (Eisner, 2003; Zhang *et al.*, 2007 & 2008; Liu *et al.*, 2009). However, one of the major challenges is that all the current models only allow one-to-one mapping from one source frontier non-terminal node (Galley *et al.*, 2004) to one target frontier non-terminal node in a bilingual translation rule. Therefore, all those translation equivalents with one-to-many frontier non-terminal node mapping cannot be covered by the current state-of-the-art models. This may largely compromise the modeling ability of translation rules.

For the second problem, currently, the combination is driven by only the source side (both tree-to-string model and tree-to-tree model only check the source span compatibility when combining different target structures in decoding) or only the

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 440–450,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

target side (string to tree model). There is no well study in considering both the source side information and the compatibility between different target syntactic structures during combination. In addition, it is well known that the traditional tree-to-tree models suffer heavily from the data sparseness issue in training and the spurious-ambiguity translation path issue (the same translation with different syntactic structures) in decoding.

In addition, because of the performance limitation of automatic syntactic parser, researchers propose using packed forest (Tomita, 1987; Klein and Manning, 2001; Huang, 2008)[1] instead of 1-best parse tree to carry out training (Mi and Huang, 2008) and decoding (Mi *et al.*, 2008) in order to reduce the side effect caused by parsing errors of the one-best tree. However, when we apply the tree-to-tree model to the bilingual forest structures, both training and decoding become very complicated.

In this paper, to address the first issue, we propose a framework to model the non-isomorphic translation process from source tree fragment to target tree sequence, allowing any one source frontier non-terminal node to be translated into any number of target frontier non-terminal nodes. For the second issue, we propose a technology to model the combination task by considering both sides' syntactic structure information. We evaluate and integrate the two technologies into forest-based tree to tree sequence translation. Experimental results on the NIST-2003 and NIST-2005 Chinese-English translation tasks show that our methods significantly outperform the forest-based tree to string and previous tree to tree models as well as the phrase-based model.

The remaining of the paper is organized as following. Section 2 reviews the related work. In section 3 and section 4, we discuss the proposed forest-based rule extraction (non-isomorphic mapping) and decoding algorithms (target syntax information usage). Finally we report the experimental results in section 5 and conclude the paper in section 6.

2 Related Work

Much effort has been done in the syntax-based translation modeling. Yamada and Knight (2001) propose a string to tree model. Galley *et al.* (2004) propose the GHKM scheme to model the string-to-tree mapping. Liu *et al.* (2006) propose a tree-to-string translation model. Liu *et al.* (2007) propose the tree sequence to string model to capture rules covered by continuous sequence of trees. Shieber (2007), DeNeefe and Knight (2009) and Carreras and Collins (2009) propose synchronous tree adjoin grammar to capture more tree-string mapping beyond the GHKM scheme. Zhang *et al.* (2009a) propose the concept of virtual node to reform a tree sequence as a tree, and design efficient algorithms for tree sequence model in forest context. All these works only consider either the source side or the target side syntax information.

To capture both side syntax contexts, Eisner (2003) studies the bilingual dependency tree-to-tree mapping in conceptual level. Zhang *et al.* (2008) propose tree sequence-based tree-to-tree modeling. Liu *et al.* (2009) propose efficient algorithms for tree-to-tree model in the forest-based training and decoding scheme. One common limitation of the above works is they only allow the one-to-one mapping between each non-terminal frontier node, and thus they suffer from the issue of rule coverage. On the other hand, due to the data sparseness issue and model coverage issue, previous tree-to-tree (Zhang *et al.*, 2008; Liu *et al.*, 2009) decoder has to rely solely on the span information or source side information to combine the target syntactic structures, without checking the compatibility of the merging nodes, in order not to fail many translation paths. Thus, this solution fails to effectively utilize the target structure information.

To address this issue, tree sequence (Liu *et al.*, 2007; Zhang *et al.*, 2008) and virtual node (Zhang *et al.*, 2009a) are two concepts with promising results reported. In this paper, with the help of these two concepts, we propose a novel framework to solve the one-to-many non-isomorphic mapping issue. In addition, our proposed solution of using target syntax information enables our forest-based tree-to-tree sequence translation decoding algorithm to not only capture bilingual forest information but also have almost the same complexity as forest-based tree-to-string translation. This reduces the time/space complexity exponentially.

3 Tree to Tree Sequence Rules

The motivation of introducing tree to tree sequence rules is to add target syntax information to tree-to-string rules. Following, we first briefly review the definition of tree-to-string rules, and then describe the tree-to-tree sequence rules.

[1] A packed forest is a compact representation of a set of trees with sharing substructures; formally, it is defined as a triple a triple$< V,E,S >$, where V is non-terminal node set, E is hyper-edge set and S is leaf node set (i.e. all sentence words). Every node in V covers a consecutive sequence of leaf, every hyper-edge in E connect the father node to its children nodes as in a tree. Figure 8 is a packed forest contains two trees.

3.1 Tree to String Rules

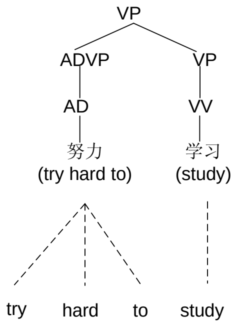


Fig. 1. A word-aligned sentence pair with source tree

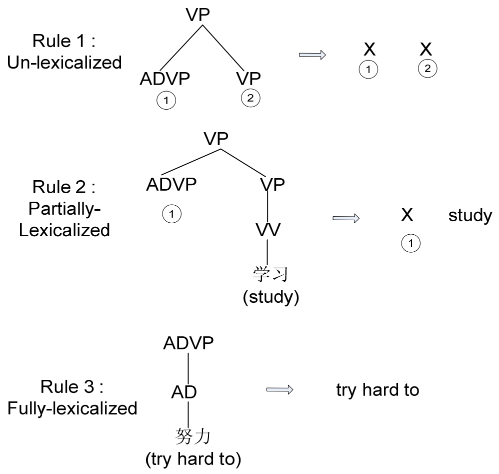


Fig. 2 Examples of tree to string rules

Fig. 2 illustrates the examples of tree to string rules extracted from Fig. 1. The tree-to-string rule is very simple. Its source side is a sub-tree of source parse tree and its target side is a string with only one variable/non-terminal X. The source side and the target side is translation of each other with the constraint of word alignments. Please note that there is no any target syntactic or linguistic information used in the tree-to-string model.

3.2 Tree to Tree Sequence Rules

It is more challenging when extracting rules with target tree structure as constraint. Fig. 3 extends Fig. 1 with target tree structure. The problem is that, given a source tree node, we are able to find its target string translation, but these target string may not form a linguistic sub-tree. For example, in Fig. 3, the source tree node "ADVP" in solid eclipse is translated to "try hard to" in the target sentence, but there is no corresponding sub-tree covering and only covering it in the target side.

Given the example rules in Fig. 2, what are their corresponding rules with target syntax information? The answer is that the previous tree or tree sequence-based models fail to model the Rule 1 and Rule 2 at Fig. 2, since at frontier node level they only allow one-to-one node mapping but the solution is one-to-many non-terminal frontier node mapping. The concept of "virtual node" (Zhang *et al.* 2009a) is a solution to this issue. To facilitate discussion, we first introduce three concepts.

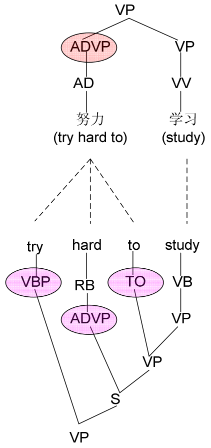


Fig. 3. A word-aligned bi-parsed tree

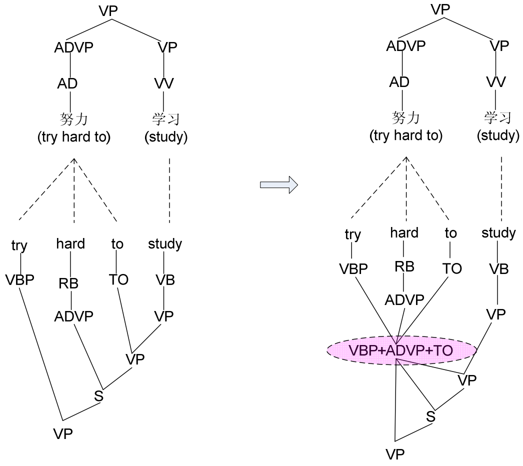


Fig. 4. A restructured tree with a virtual span root

- **Def. 1**. The ***"node sequence"*** is a sequence of nodes (either leaf or internal nodes) covering a consecutive span. For example, in Fig 3, "VBP RB TO" and "VBP ADVP TO" are two "node sequence" covering the same span "try hard to".

- **Def. 2**. The "***root node sequence***" of a span is such a node sequence that any node in this sequence could not be a child of a node in other node sequence of the span. Intuitively, the "*root node sequence*" of a span is the node sequence with the highest topology level. For example, "VBP ADVP TO" is the "root node sequence" of the span of "try hard to". It is easy to prove that given any span, there exist one and only one "*root node sequence*".

- **Def. 3**. The "***span root***" of a span is such a node that if the "*root node sequence*" contains only one tree node, then the "*span root*" is this tree node; otherwise, the "*span root*" is the virtual father node (Zhang *et al*., 2009a) of the "*root node sequence*". Fig. 4 illustrates the reformed Fig. 3 by introducing the virtual node "VBP+ADVP+TO" as the "span root" of the span of "try hard to".

The "*span root*" facilitates us to extract rules with target side structure information. Given a sub-tree of the source tree, we have a set of non-terminal frontier nodes. For each such frontier node, we can find its corresponding target "*span root*". If the "*span root*" is a virtual node, then we add it into the target tree as a virtual segmentation joint point. After adding the "span root" as joint point, we are able to ensure that each frontier source node has only one corresponding target node, then we can use any traditional rule extraction algorithm to extract rules, including those rules with one-to-many non-terminal frontier mappings.

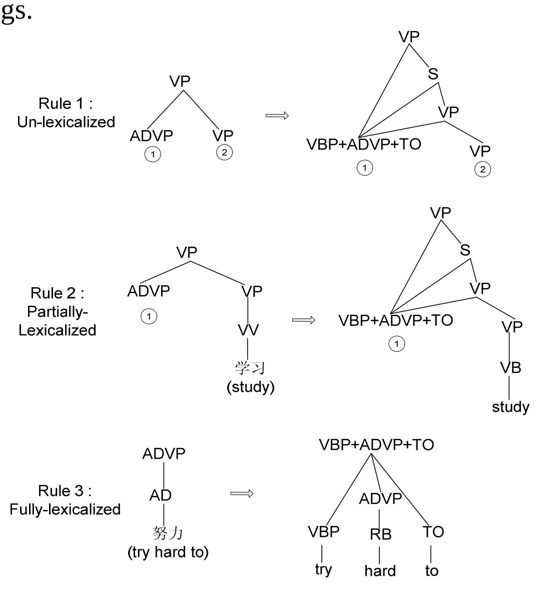


Fig. 5. Tree-to-tree sequence rules

Fig. 5 lists the corresponding rules with target structure information of the tree-to-string rules in Fig 2. All the three rules cannot be extracted by previous tree-to-tree mapping methods (Liu *et al*., 2009). The previous tree-sequence-based methods (Zhang *et al*., 2008; Zhang *et al*., 2009a) can extracted rule 3 since they allow one-to-many mapping in root node level. But they cannot extract rule 1 and rule 2. Therefore, for any tree-to-string rule, our method can always find the corresponding tree-to-tree sequence rule. As a result, our rule coverage is the same as tree-to-string framework while our rules contain more informative target syntax information. Later we will show that using our decoding algorithm the tree-to-tree sequence search space is exponentially reduced to the same as tree-to-string search space. That is to say, we do not need to worry about the exponential search space issue of tree-to-tree sequence model existing in previous work.

3.3 Rule Extraction in Tree Context

Given a word aligned tree pair, we first extract the set of minimum tree to string rules (Galley *et al.* 2004), then for each tree-to-string rule, we can easily extract its corresponding tree-to-tree sequence rule by introducing the virtual span root node. After that, we generate the composite rules by iteratively combining small rules.

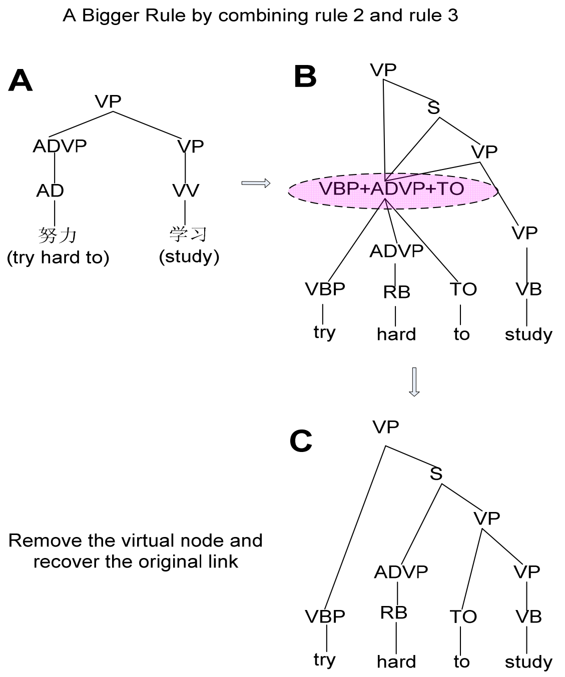


Fig. 6. Rule combination and virtual node removing

Please note that in generating composite rules, if the joint node is a virtual node, we have to recover the original link and remove this virtual node to avoid unnecessary ambiguity. Fig. 6 illustrates the combination process of rule 2 and rule 3 in Fig. 5. As a result, all of our extract rules do not contain any internal virtual nodes.

3.4 Rule Extraction in Forest Context

In forest pair context, we also first generate the minimum tree-to-string rule set as Mi *et al.* (2008), and for each tree-to-string rule, we find its corresponding tree-to-tree sequence rules, and then do rule composition.

In tree pair context, given a tree-to-string rule, there is one and only one corresponding tree-to-tree sequence rule. But in forest pair context, given one such tree-to-string rule, there are many corresponding tree-to-tree sequence rules. All these sub-trees form one or more sub-forests[2] of the entire big target forest. If we can identify the sub-forests, i.e., all of the hyper-edges of the sub-forests, we can retrieve all the sub-trees from the sub-forests as the target sides of the corresponding tree-to-tree sequence rules.

Given a source sub-tree, we can obtain the target root span where the target sub-forests start and the frontier spans where the target sub-forests stop. To indentify all the hyper-edges in the sub-forests, we start from every node covering the root span, traverse from top to down, mark all the hyper-edges visited and stop at the node if its span is a sub-span of one of the forest frontier spans or if it is a word node. The reason we stop at the node once it fell into a frontier span (i.e. the span of the node is a sub-span of the frontier span) is to guarantee that given any frontier span, we could stop at the "*root node sequence*" of this span by **Def.** 2.

For example, Fig. 7 is a source sub-tree of rule 2 in Fig. 5 and the circled part in Fig. 8 is one of its corresponding target sub-forests. Its corresponding target root span is [1,4] (corresponding to source root "VP") and its corresponding target frontier span is {[1,3], study[4,4]}. Now given the target forest, we start from node VP[1,4] and traverse from top to down, finally stop at following nodes: VBP[1,1], ADVP[2,2], TO[3,3], study .

Please note that the starting root node must be a single node, being either a normal forest node or a virtual "span root" node. The virtual "span root" node serves as the frontier node of upper rules and root node of the currently being extracted rules. Because we extract rules in a top-to-down manner, the necessary virtual "span root" node for current sub-forest has already been added into the global forest when extracting upper level rules.

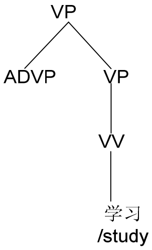


Figure 7. A source sub-tree in rule 2

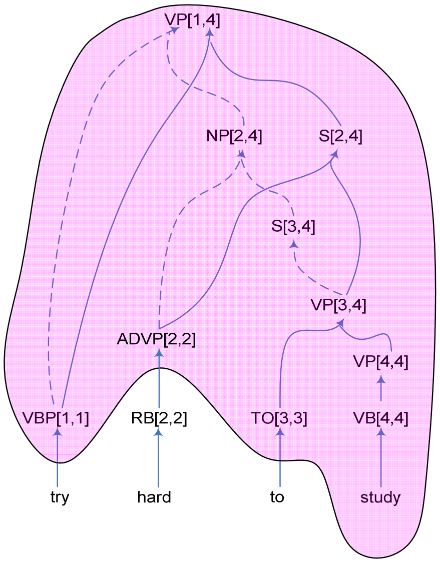


Fig. 8. The corresponding target sub-forest for the tree of Figure 7.

3.5 Fractional Count of Rule

Following Mi and Huang (2008) and Liu *et al.* (2009), we assign a fractional count to a rule to measure how likely it appears given the context of the forest pair. In following equation, "S" means source sub-tree, "T" means target sub-tree, "SF" is source forest and "TF" is the target forest.

$$P(S,T\,|SF,TF) \cong P(S|SF,TF) * P(T|SF,TF)$$
$$\cong P(S|SF) * P(T|TF)$$

[2] All the sub-forests cover the same span. But their roots have different grammar tags as the roots' names. The root may be a virtual span root node in the case of the one-to-many frontier non-terminal node mappings.

The above equation means the fractional count of a source-target tree pair is just the product of each of their fractional count in corresponding forest context in following equation.

$$P(subtree\ |Forest) = \frac{\alpha\beta(subtree)}{\alpha\beta(forest\ root)}$$
$$= \frac{\alpha(subtree\ root) * \prod_{h \in subtree} P(h) * \prod_{v \in leaves(subtree)} \beta(v)}{\alpha\beta(forest\ root)}$$

where α and β are the outside and inside probabilities. In addition, if a sub-tree root is a virtual node (formed by a root node sequence), then we use following equation to approximate the outside probability of the virtual node.

$$\alpha(node\ sequence\ ns) = \sqrt[\#\ of\ nodes\ in\ ns]{\prod_{n \in ns} \alpha(n)}$$

4 Decoding

4.1 Traditional Forest-based Decoding

A typical translation process of a forest-based system is to first convert the source packed forest into a target translation forest, and then apply search algorithm to find the best translation result from this target translation forest (Mi *et al.*, 2008).

For the tree-to-string model, the forest conversion process is as following: given an input packed forest, we do pattern matching (Zhang *et al.*, 2009b) with the source side structures in the rule set. For each matched rule, we establish its target side as a hyper-edge in the target forest.

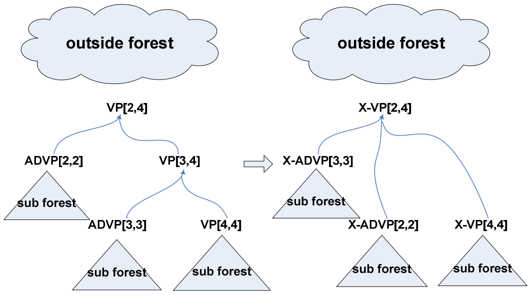


Fig. 9. A forest conversion step in a tree to string model

Fig. 9 exemplifies a conversion step in the tree to string model. A sub-tree structure with two hyper-edge "*VP[2,4] => ADVP[2,2] VP[3,4]*" and "*VP[3,4] => ADVP[3,4] VP[4,4]*" is converted into a target hyper-edge "*X-VP[2,4] => X-ADVP[3,3] X-ADVP[2,2] X-VP[4,4]* ". The node "*X-VP[4,4]*" in the target forest means that its syntactic label in target forest is "X" and it is translated from the source node "VP[4,4]" in the source forest. In this target hyper-edge, "*X-ADVP[3,3] X-ADVP[2,2]*" means the translation from source node "*ADVP[3,3]*" is put before the translation from "*ADVP[2,2]*", representing a structure reordering.

4.2 Toward Bilingual Syntax-aware Translation Generation

As we could see in section 4.1, there is only one kind of non-terminal symbol "X" in the target side. It is a big challenge to rely on such a coarse label to generate a translation with fine syntactic quality. For example, a source node may be translated into a "NP" (noun phrase) in target side. However, in this rule set with the only symbol "X", it may be merged with upper structure as a "VP" (verb phrase) instead, because there is no way to favor one over another. In this case, the target tree does not well model the translation syntactically. In addition, all of the internal structure information in the target side is ignored by the tree-to-string rules.

One natural solution to the above issue is to use the tree to tree/tree sequence model, which have richer target syntax structures for more discriminative probability and finer labels to guide the combination process. However, the tree to tree/tree sequence model may face very severe computational problem and so-called "spurious ambiguities" issue.

Theoretically, if in the tree-to-tree sequence model-based decoding, we just give a penalty to the incompatible-node combinations instead of pruning out the translation paths, then the set of sentences generated by the tree-to-tree sequence model is identical to that of the tree-to-string model since every tree-to-tree sequence rule can be projected into a tree-to-string rule. Motivated by this, we propose a solution call parallel hypothesis spaces searching to solve the computational and "spurious ambiguities" issues mentioned above. In the meanwhile, we can fully utilize the target structure information to guide translation.

We restructure the tree-to-tree sequence rule set by grouping all the rules according to their corresponding tree-to-string rules. This behaves like a "tree-to-forest" rule. The "forest" encodes all the tree sequences with same corresponding string. With the re-constructed rule set, during decoding, we generate two target translation hypothesis spaces (in the form of packed forests) synchronously by the tree-to-string rules and tree-to-tree sequence rules, and maintain the projection between them. In other words, we generate hypothesis (searching) from the tree-to-string forest and calculate the probability (evaluating syntax goodness) for each hypothesis by the hyper-edges in the tree-to-tree sequence forest.

4.3 Parallel Hypothesis Spaces

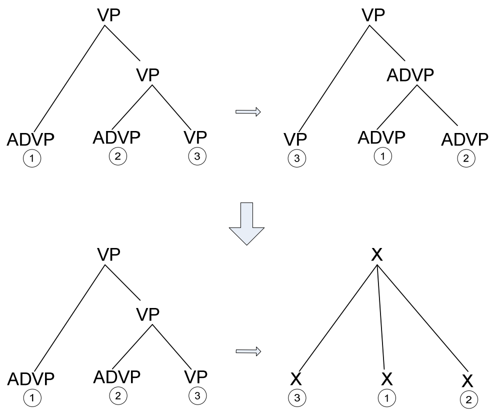


Fig. 10. Mapping from tree-to-tree sequence into tree-to-string rule

In this subsection, we describe what the parallel search spaces are and how to construct them. As shown at Fig. 10, given a tree-to-tree sequence rule, it is easy to find its corresponding tree-to-string rule by simply ignoring the target inside structure and renaming the root and leaves non-terminal labels into “X”. We iterate through the tree-to-tree sequence rule set, find its corresponding tree-to-string rule and then group those rules with the same tree-to-string projection. After that, the original tree-to-tree sequence rule set becomes a set of smaller rule sets. Each of them is indexed by a unique tree-to-string rule.

We apply the tree-to-string rules to generate an explicit target translation forest to represent the target sentences space. At the same time, whenever a tree-to-string rule is applied, we also retrieve its corresponding tree-to-tree sequence rule set and generate a set of latent hyper-edges with fine-grained syntax information. In this case, we have two parallel forests, one with coarse explicit hyper-edges and the other fine and latent. Given a hyper-edge (or a node) in the coarse forest, there are a group of corresponding latent hyper-edges (or nodes) with finer syntax labels in the fine forest. Accordingly, given a tree in the coarse forest, there is a corresponding sub-forest in the latent fine forest. We can view the latent fine forest as imbedded inside the explicit coarse forest. If an explicit hyper-edge is viewed as a big cable, then the group of its corresponding latent hyper-edges is the small wires inside it.

We rely on the explicit hyper-edges to enumerate possible hypothesis while using the latent hyper-edges to measure its translation probability and syntax goodness. Thus, the complexity of the search space is reduced into the tree-to-string model level, while keeping the target language generation syntactic aware. More importantly, we thoroughly avoid those spurious ambiguities introduced by the tree-to-tree sequence rules.

4.4 Decoding with Parallel Hypothesis Spaces

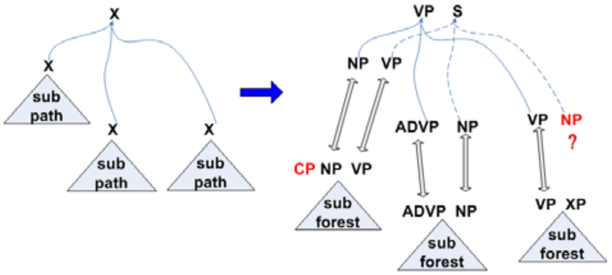


Fig. 11. Derivation path and derivation forest

In this subsection, we show exactly how our decoder finds the best result from the parallel spaces. We generate hypothesis by traversing the coarse forest in the parallel spaces with cube-pruning (Huang and Chiang, 2007). Given a newly generated hypothesis, it is affiliated with a derivation path (tree) in the coarse forest and a group of derivation paths (sub-forest) in the finer forest. As shown in Fig. 11, the left part is the derivation path formed by a coarse hyper-edge, consisting the newly-generated sub-tree “X => X X X” connecting with three previously-generated sub paths while the right part is the derivation forest formed by newly-generated finer hyper-edges rooted at “VP” and “S”, and previously-generated sub-forests.

In this paper, we use the sum of probabilities of all the derivation paths in the finer forest to measure the quality of the candidate translation suggested by the hypothesis. From Fig. 11, we can see there may be more than one corresponding finer forests, it is easy to understand that the sum of all the trees’ probabilities in these finer forests is equal to the sum of the inside probability of all these root nodes of these finer forests. We adopt the dynamic programming to compute the probability of the finer forest: whenever we generate a new hypothesis by concatenating a coarse hyper-edge and its sub-path, we find its corresponding finer hyper-edges and sub-forests, do the combination and accumulate probabilities from bottom to up. For the coarse hyper-edge, because there is only one label "X", any sub-path could be easily concatenated with upper structure covering the same sub-span without the need of checking label compatibility. While for the finer hyper-edges, we only link the root nodes of sub-forests to upper hyper-edges with the same linking node label. This is to guarantee syntactic goodness. In case there are some leaf nodes of the upper hyper-edges fail to find corresponding sub-forest roots with the same label (e.g. the "NP" in red color in the rightmost of Fig 11), we simply link it into the nodes with the least inside probability (among these sub-forests), and at the same time give a penalty score to this combination. If some root nodes of some sub-forest still cannot find upper leaf nodes to concatenate (e.g. the "CP" in red color in Fig. 11), we simply ignore them. After the combination process, it is straightforward to accumulate the inside probability dynamically from bottom up.

5 Experiment

5.1 Experimental Settings

We evaluate our method on the Chinese-English translation task. We first carry out a series empirical study on a set of parallel data with 30K sentence pairs, and then do experiment on a larger data set to ensure that the effectiveness of our method is consistent across data set of different size. We use the NIST 2002 test set as our dev set, and NIST 2003 and NIST 2005 test sets as our test set. A 3-gram language model is trained on the target side of the training data by the SRILM Toolkits (Stolcke, 2002) with modified Kneser-Ney smoothing (Kneser and Ney, 1995). We train Charniak's parser (Charniak, 2000) on CTB5.0 for Chinese and ETB3.0 for English and modify it to output packed forest. GIZA++ (Och and Ney, 2003) and the heuristics "grow-diag-final-and" are used to generate *m-to-n* word alignments. For the MER training (Och, 2003), Koehn's MER trainer (Koehn, 2007) is modified for our system. For significance test, we use Zhang *et al.*'s implementation (Zhang *et al*, 2004). Our evaluation metrics is *case-sensitive closest* BLEU-4 (Papineni *et al.*, 2002). We use following features in our systems: 1) bidirectional tree-to-tree sequence probability, 2) bidirectional tree-to-string probability, 3) bidirectional lexical translation probability, 4) target language model, 5) source tree probability 6) the average number of unmatched nodes in the target forest. 7) the length of the target translation, 8) the number of glue rules used.

5.2 Empirical Study on Small Data

We set forest pruning threshold (Mi *et al*., 2008) to 8 on both source and target forests for rule extraction. For each source sub-tree, we set its height up to 3, width up to 7 and extract up to 10-best target structures. In decoding, we set the pruning threshold to 10 for the input source forest. Table 1 compares the performance in NIST 2003 data set of our method and several state-of-the-art systems as our baseline.

1) **MOSES**: phrase-based system (Koehn *et al*., 2007)
2) **FT2S**: forest-based tree-to-string system (Mi and Huang, 2008; Mi *et al*., 2008)
3) **FT2T**: forest-based tree-to-tree system (Liu *et al*., 2009).
4) **FT2TS (1to1)**: our forest-based tree-to-tree sequence system, where 1to1 means only one-to-one frontier non-terminal node mapping is allowed, thus the system does not follow our non-isomorphic mapping framework.
5) **FT2TS (1toN)**: our forest-based tree-to-tree sequence system that allows one-to-many frontier non-terminal node mapping by following our non-isomorphic mapping framework

In addition, our proposed parallel searching space (**PSS**) technology can be applied to both tree to tree and tree to sequence systems. Thus in table 1, for the tree-to-tree/tree sequence systems, we report two BLEU scores, one uses this technology (withPSS) and one does not (noPSS).

Model		**BLEU-4**
MOSES		23.39
FT2S		26.10
FT2T	noPSS	23.40
	withPSS	24.46
FT2TS (1to1)	noPSS	25.39
	withPSS	26.58
FT2TS (1toN)	noPSS	26.30
	withPSS	**27.70**

Table 1. Performance comparison of different methods

From Table 1, we can see that:

1) All the syntax-based systems (except FT2T (noPSS) (23.40)) consistently outperform the phrase-based system MOSES significantly ($p < 0.01$), indicating that syntactic knowledge is very useful to SMT.
2) The PSS technology shows significant performance improvement ($p < 0.01$) in all models, which clearly shows effectiveness of the PSS technology in utilizing target structures for target language generation.
3) FT2TS (1toN) significantly outperforms ($p < 0.01$) FT2TS (1to1) in both cases (noPSS and withPSS). This convincingly shows the effectiveness of our non-isomorphic mapping framework in capturing the non-isomorphic structure translation equivalences.
4) Both FT2TS systems significantly outperform FT2T($p < 0.01$). This verifies the effectiveness of tree sequence rules.
5) FT2TS shows different level of performance improvements over FT2S with the best case having 1.6 (27.70-26.10) BLEU score improvement over FT2S. This suggests that the target structure information is very useful, but we need to find a correct way to effectively utilize it.

1to1	1toN	ratio
1735871	2363771	1:1.36

Table 2. Statistics on node mapping in forest, where “1to1” means the number of nodes in source forest that can be translated into one node in target forest and “1toN” means the number of nodes in source forest that have to be translated into more than one node in target forest, where the node refers to non-terminal nodes only

Model	# of rules	T2S covered
FT2T	295732	26.8%
FT2TS(1to1)	631487	57.1%
FT2TS (1toN)	1945168	100%

Table 3. Statistics of rule coverage, where “T2S covered” means the percentage of tree-to-string rules that can be covered by the model

Table 2 studies the node isomorphism between bilingual forest pair. We can see that the non-isomorphic node translation mapping (1toN) accounts for 57.6% (=1.36/(1+1.36)) of all the forest non-terminal nodes with target translation. This means that the one-to-many node mapping is a major issue in structure transformation. It also empirically justifies the importance of our non-isomorphic mapping framework.

Table 3 shows the rule coverage of different bilingual structure mapping model. FT2T only covers 26.8% tree-to-string rules, so it performs worse than FT2S as shown in Table 1. FT2TS (1to1) does not allow one-to-many frontier node mapping, so it could only recover the non-isomorphic node mapping in the root level, while FT2TS (1toN) could make it at both root and leaf levels. Therefore, it is not surprising that in Table 3, FT2TS (1toN) cover many more rules than FT2TS (1to1) because given a source tree, there are many leaves, if any one of them is non-isomorphic, then it could not be covered by the FT2TS (1to1).

Decoding Method	BLEU-4	Speed (sec/sent)
Traditional: FT2TS (1toN) (**noPPS**)	26.30	152.6
Ours: FT2TS (1toN) (**withPPS**)	**27.70**	5.22

Table 4. Performance and speed comparison

Table 4 clearly shows the advantage of our decoder over the traditional one. Ours could not only generate better translation result, but also be 152.6/5.22>30 times faster. This mainly attributes to two reasons: 1) one-to-many frontier node mapping equipments the model with more ability to capture more non-isomorphic structure mappings than traditional models, and 2) “parallel search space” enables the decoder to fully utilize target syntactic information, but keeping the size of search space the same as that a “tree to string” model explores.

5.3 Results on Larger Data Set

We also carry out experiment on a larger dataset consisting of the small dataset used in last section and the FBIS corpus. In total, there are 280K parallel sentence pairs with 9.3M Chinese words and 11.8M English words. A 3-gram language model is trained on the target side of the parallel corpus and the GIGA3 Xinhua portion. We compare our system (FT2TS with 1toN and withPPS) with two state-of-the-art baselines: the phrase-based system MOSES and the forest-based tree-to-string system implemented by us. Table 5 clearly shows the effectiveness of our method is consistent across small and larger corpora, outperforming FT2S by 1.6-1.8 BLEU and the MOSES by 3.3-4.0 BLEU statistically significantly (p<0.01).

Model	**BLEU**	
	NIST2003	**NIST2005**
MOSES	29.51	27.53
FT2S	31.21	29.72
FT2TS	**32.88**	**31.50**

Table 5. Performance on larger data set

6 Conclusions

In this paper, we propose a framework to address the issue of bilingual non-isomorphic structure mapping and a novel parallel searching space scheme to effectively utilize target syntactic structure information in the context of forest-based tree to tree sequence machine translation. Based on this framework, we design an efficient algorithm to extract tree-to-tree sequence translation rules from word aligned bilingual forest pairs. We also elaborate the parallel searching space-based decoding algorithm and the node label checking scheme, which leads to very efficient decoding speed as fast as the forest-based tree-to-string model does, at the same time is able to utilize informative target structure knowledge. We evaluate our methods on both small and large training data sets and two NIST test sets. Experimental results show our methods statistically significantly outperform the state-of-the-art models across different size of corpora and different test sets. In the future, we are interested in testing our algorithm at forest-based tree sequence to tree sequence translation.

References

Eugene Charniak. 2000. *A maximum-entropy inspired parser*. NAACL-00.

Eugene Charniak, Kevin Knight, and Kenji Yamada. 2003. *Syntax-based language models for statistical machine translation.* MT Summit IX. 40–46.

David Chiang. 2007. *Hierarchical phrase-based translation.*Computational Linguistics, 33(2).

Steve DeNeefe, Kevin Knight. 2009. *Synchronous Tree Adjoining Machine Translation*. EMNLP-2009. 727-736.

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. 273-280.

Liang Huang. 2008. Forest Reranking: *Discriminative Parsing with Non-Local Features*. ACL-HLT-08. 586-594

Liang Huang and David Chiang. 2005. *Better k-best Parsing*. IWPT-05. 53-64

Liang Huang and David Chiang. 2007. *Forest rescoring: Faster decoding with integrated language models*. ACL-07. 144–151

Dan Klein and Christopher D. Manning. 2001. *Parsing and Hypergraphs*. IWPT-2001.

Reinhard Kneser and Hermann Ney. 1995. *Improved backing-off for M-gram language modeling*. ICASSP-95, 181-184

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. 177-180. (poster)

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.

Yang Liu, Yajuan Lü, Qun Liu. 2009. *Improving Tree-to-Tree Translation with Packed Forests.* ACL-09. 558-566

Haitao Mi, Liang Huang, and Qun Liu. 2008. *Forest-based translation*. ACL-HLT-08. 192-199.

Haitao Mi and Liang Huang. 2008. *Forest-based Translation Rule Extraction*. EMNLP-08. 206-214.

Franz J. Och. 2003. *Minimum error rate training in statistical machine translation.* ACL-03. 160-167.

Franz Josef Och and Hermann Ney. 2003. *A Systematic Comparison of Various Statistical Alignment Models*. Computational Linguistics. 29(1) 19-51.

Kishore Papineni, Salim Roukos, Todd Ward and Wei-Jing Zhu. 2002. *BLEU: a method for automatic evaluation of machine translation.* ACL-02. 311-318.

Andreas Stolcke. 2002. *SRILM - an extensible language modeling toolkit.* ICSLP-02. 901-904.

Masaru Tomita. 1987. *An Efficient Augmented-Context-Free Parsing Algorithm*. Computational Linguistics 13(1-2): 31-46

Xavier Carreras and Michael Collins. 2009. *Non-projective Parsing for Statistical Machine Translation*. EMNLP-2009. 200-209.

K. Yamada and K. Knight. 2001. *A Syntax-Based Statistical Translation Model.* ACL-01. 523-530.

Hui Zhang, Min Zhang, Haizhou Li, Aiti Aw and Chew Lim Tan. 2009a. *Forest-based Tree Sequence to String Translation Model.* ACL-IJCNLP-09. 172-180.

Hui Zhang, Min Zhang, Haizhou Li, and Chew Lim Tan. 2009b. *Fast Translation Rule Matching for Syntax-based Statistical Machine Translation.* EMNLP-09. 1037-1045.

Min Zhang, Hongfei Jiang, Ai Ti Aw, Jun Sun, Chew Lim Tan and Sheng Li. 2007. *A Tree-to-Tree Alignment-based model for statistical Machine translation*. MT-Summit-07. 535-542

Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan, Sheng Li. 2008. *A Tree Sequence Alignment-based Tree-to-Tree Translation Model.* ACL-HLT-08. 559-567.

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.

Discriminative Instance Weighting for Domain Adaptation in Statistical Machine Translation

George Foster and **Cyril Goutte** and **Roland Kuhn**
National Research Council Canada
283 Alexandre-Taché Blvd
Gatineau, QC J8X 3X7
first.last@nrc.gc.ca

Abstract

We describe a new approach to SMT adaptation that weights out-of-domain phrase pairs according to their relevance to the target domain, determined by both how similar to it they appear to be, and whether they belong to general language or not. This extends previous work on discriminative weighting by using a finer granularity, focusing on the properties of instances rather than corpus components, and using a simpler training procedure. We incorporate instance weighting into a mixture-model framework, and find that it yields consistent improvements over a wide range of baselines.

1 Introduction

Domain adaptation is a common concern when optimizing empirical NLP applications. Even when there is training data available in the domain of interest, there is often additional data from other domains that could in principle be used to improve performance. Realizing gains in practice can be challenging, however, particularly when the target domain is distant from the background data. For developers of Statistical Machine Translation (SMT) systems, an additional complication is the heterogeneous nature of SMT components (word-alignment model, language model, translation model, etc.), which precludes a single universal approach to adaptation.

In this paper we study the problem of using a parallel corpus from a background domain (OUT) to improve performance on a target domain (IN) for which a smaller amount of parallel training material—though adequate for reasonable performance—is also available. This is a standard adaptation problem for SMT. It is difficult when IN and OUT are dissimilar, as they are in the cases we study. For simplicity, we assume that OUT is homogeneous. The techniques we develop can be extended in a relatively straightforward manner to the more general case when OUT consists of multiple sub-domains.

There is a fairly large body of work on SMT adaptation. We introduce several new ideas. First, we aim to explicitly characterize examples from OUT as belonging to general language or not. Previous approaches have tried to find examples that are similar to the target domain. This is less effective in our setting, where IN and OUT are disparate. The idea of distinguishing between general and domain-specific examples is due to Daumé and Marcu (2006), who used a maximum-entropy model with latent variables to capture the degree of specificity. Daumé (2007) applies a related idea in a simpler way, by splitting features into general and domain-specific versions. This highly effective approach is not directly applicable to the multinomial models used for core SMT components, which have no natural method for combining split features, so we rely on an instance-weighting approach (Jiang and Zhai, 2007) to downweight domain-specific examples in OUT. Within this framework, we use features intended to capture degree of generality, including the output from an SVM classifier that uses the intersection between IN and OUT as positive examples.

Our second contribution is to apply instance

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 451–459,
MIT, Massachusetts, USA, 9-11 October 2010.

weighting at the level of phrase pairs. Sentence pairs are the natural instances for SMT, but sentences often contain a mix of domain-specific and general language. For instance, the sentence *Similar improvements in haemoglobin levels were reported in the scientific literature for other epoetins* would likely be considered domain-specific despite the presence of general phrases like *were reported in*. Phrase-level granularity distinguishes our work from previous work by Matsoukas et al (2009), who weight sentences according to sub-corpus and genre membership.

Finally, we make some improvements to baseline approaches. We train linear mixture models for conditional phrase pair probabilities over IN and OUT so as to maximize the likelihood of an empirical joint phrase-pair distribution extracted from a development set. This is a simple and effective alternative to setting weights discriminatively to maximize a metric such as BLEU. A similar maximum-likelihood approach was used by Foster and Kuhn (2007), but for language models only. For comparison to information-retrieval inspired baselines, eg (Lü et al., 2007), we select sentences from OUT using language model perplexities from IN. This is a straightforward technique that is arguably better suited to the adaptation task than the standard method of treating representative IN sentences as queries, then pooling the match results.

The paper is structured as follows. Section 2 describes our baseline techniques for SMT adaptation, and section 3 describes the instance-weighting approach. Experiments are presented in section 4. Section 5 covers relevant previous work on SMT adaptation, and section 6 concludes.

2 Baseline SMT Adaptation Techniques

Standard SMT systems have a hierarchical parameter structure: top-level log-linear weights are used to combine a small set of complex features, interpreted as log probabilities, many of which have their own internal parameters and objectives. The top-level weights are trained to maximize a metric such as BLEU on a small development set of approximately 1000 sentence pairs. Thus, provided at least this amount of IN data is available—as it is in our setting—adapting these weights is straightforward. We focus here instead on adapting the two most important features: the language model (LM), which estimates the probability $p(w|h)$ of a target word w following an ngram h; and the translation models (TM) $p(s|t)$ and $p(t|s)$, which give the probability of source phrase s translating to target phrase t, and vice versa. We do not adapt the alignment procedure for generating the phrase table from which the TM distributions are derived.

2.1 Simple Baselines

The natural baseline approach is to concatenate data from IN and OUT. Its success depends on the two domains being relatively close, and on the OUT corpus not being so large as to overwhelm the contribution of IN.

When OUT is large and distinct, its contribution can be controlled by training separate IN and OUT models, and weighting their combination. An easy way to achieve this is to put the domain-specific LMs and TMs into the top-level log-linear model and learn optimal weights with MERT (Och, 2003). This has the potential drawback of increasing the number of features, which can make MERT less stable (Foster and Kuhn, 2009).

2.2 Linear Combinations

Apart from MERT difficulties, a conceptual problem with log-linear combination is that it multiplies feature probabilities, essentially forcing different features to agree on high-scoring candidates. This is appropriate in cases where it is sanctioned by Bayes' law, such as multiplying LM and TM probabilities, but for adaptation a more suitable framework is often a mixture model in which each event may be generated from some domain. This leads to a linear combination of domain-specific probabilities, with weights in $[0, 1]$, normalized to sum to 1.

Linear weights are difficult to incorporate into the standard MERT procedure because they are "hidden" within a top-level probability that represents the linear combination.[1] Following previous work (Foster and Kuhn, 2007), we circumvent this problem by choosing weights to optimize corpus log-likelihood, which is roughly speaking the training criterion used by the LM and TM themselves.

[1] This precludes the use of exact line-maximization within Powell's algorithm (Och, 2003), for instance.

For the LM, adaptive weights are set as follows:

$$\hat{\alpha} = \underset{\alpha}{\operatorname{argmax}} \sum_{w,h} \tilde{p}(w,h) \log \sum_i \alpha_i p_i(w|h), \quad (1)$$

where α is a weight vector containing an element α_i for each domain (just IN and OUT in our case), p_i are the corresponding domain-specific models, and $\tilde{p}(w,h)$ is an empirical distribution from a target-language training corpus—we used the IN dev set for this.

It is not immediately obvious how to formulate an equivalent to equation (1) for an adapted TM, because there is no well-defined objective for learning TMs from parallel corpora. This has led previous workers to adopt ad hoc linear weighting schemes (Finch and Sumita, 2008; Foster and Kuhn, 2007; Lü et al., 2007). However, we note that the final conditional estimates $p(s|t)$ from a given phrase table maximize the likelihood of joint empirical phrase pair counts over a word-aligned corpus. This suggests a direct parallel to (1):

$$\hat{\alpha} = \underset{\alpha}{\operatorname{argmax}} \sum_{s,t} \tilde{p}(s,t) \log \sum_i \alpha_i p_i(s|t), \quad (2)$$

where $\tilde{p}(s,t)$ is a joint empirical distribution extracted from the IN dev set using the standard procedure.[2]

An alternative form of linear combination is a *maximum a posteriori* (MAP) combination (Bacchiani et al., 2004). For the TM, this is:

$$p(s|t) = \frac{c_I(s,t) + \beta\, p_o(s|t)}{c_I(t) + \beta}, \quad (3)$$

where $c_I(s,t)$ is the count in the IN phrase table of pair (s,t), $p_o(s|t)$ is its probability under the OUT TM, and $c_I(t) = \sum_{s'} c_I(s',t)$. This is motivated by taking $\beta\, p_o(s|t)$ to be the parameters of a Dirichlet prior on phrase probabilities, then maximizing posterior estimates $p(s|t)$ given the IN corpus. Intuitively, it places more weight on OUT when less evidence from IN is available. To set β, we used the same criterion as for α, over a dev corpus:

$$\hat{\beta} = \underset{\beta}{\operatorname{argmax}} \sum_{s,t} \tilde{p}(s,t) \log \frac{c_I(s,t) + \beta\, p_o(s|t)}{c_I(t) + \beta}.$$

The MAP combination was used for TM probabilities only, in part due to a technical difficulty in formulating coherent counts when using standard LM smoothing techniques (Kneser and Ney, 1995).[3]

2.3 Sentence Selection

Motivated by information retrieval, a number of approaches choose "relevant" sentence pairs from OUT by matching individual source sentences from IN (Hildebrand et al., 2005; Lü et al., 2007), or individual target hypotheses (Zhao et al., 2004). The matching sentence pairs are then added to the IN corpus, and the system is re-trained. Although matching is done at the sentence level, this information is subsequently discarded when all matches are pooled.

To approximate these baselines, we implemented a very simple sentence selection algorithm in which parallel sentence pairs from OUT are ranked by the perplexity of their target half according to the IN language model. The number of top-ranked pairs to retain is chosen to optimize dev-set BLEU score.

3 Instance Weighting

The sentence-selection approach is crude in that it imposes a binary distinction between useful and non-useful parts of OUT. Matsoukas et al (2009) generalize it by learning weights on sentence pairs that are used when estimating relative-frequency phrase-pair probabilities. The weight on each sentence is a value in $[0,1]$ computed by a perceptron with Boolean features that indicate collection and genre membership.

We extend the Matsoukas et al approach in several ways. First, we learn weights on individual phrase pairs rather than sentences. Intuitively, as suggested by the example in the introduction, this is the right granularity to capture domain effects. Second, rather than relying on a division of the corpus into manually-assigned portions, we use features intended to capture the usefulness of each phrase pair. Finally, we incorporate the instance-weighting model into a general linear combination, and learn weights and mixing parameters simultaneously.

[2] Using non-adapted IBM models trained on all available IN and OUT data.

[3] Bacchiani et al (2004) solve this problem by reconstituting joint counts from smoothed conditional estimates and unsmoothed marginals, but this seems somewhat unsatisfactory.

3.1 Model

The overall adapted TM is a combination of the form:

$$p(s|t) = \alpha_t \, p_I(s|t) + (1 - \alpha_t) \, p_o(s|t), \tag{4}$$

where $p_I(s|t)$ is derived from the IN corpus using relative-frequency estimates, and $p_o(s|t)$ is an instance-weighted model derived from the OUT corpus. This combination generalizes (2) and (3): we use either $\alpha_t = \alpha$ to obtain a fixed-weight linear combination, or $\alpha_t = c_I(t)/(c_I(t) + \beta)$ to obtain a MAP combination.

We model $p_o(s|t)$ using a MAP criterion over weighted phrase-pair counts:

$$p_o(s|t) = \frac{c_\lambda(s,t) + \gamma u(s|t)}{\sum_{s'} c_\lambda(s',t) + \gamma} \tag{5}$$

where $c_\lambda(s,t)$ is a modified count for pair (s,t) in OUT, $u(s|t)$ is a prior distribution, and γ is a prior weight. The original OUT counts $c_o(s,t)$ are weighted by a logistic function $w_\lambda(s,t)$:

$$\begin{aligned} c_\lambda(s,t) &= c_o(s,t) \; w_\lambda(s,t) \qquad (6) \\ &= c_o(s,t) \; [1 + \exp(-\sum_i \lambda_i f_i(s,t))]^{-1}, \end{aligned}$$

where each $f_i(s,t)$ is a feature intended to characterize the usefulness of (s,t), weighted by λ_i.

The mixing parameters and feature weights (collectively ϕ) are optimized simultaneously using dev-set maximum likelihood as before:

$$\hat{\phi} = \operatorname*{argmax}_{\phi} \sum_{s,t} \tilde{p}(s,t) \log p(s|t;\phi). \tag{7}$$

This is a somewhat less direct objective than used by Matsoukas et al, who make an iterative approximation to expected TER. However, it is robust, efficient, and easy to implement.[4]

To perform the maximization in (7), we used the popular L-BFGS algorithm (Liu and Nocedal, 1989), which requires gradient information. Dropping the conditioning on ϕ for brevity, and letting $\bar{c}_\lambda(s,t) = c_\lambda(s,t) + \gamma u(s|t)$, and $\bar{c}_\lambda(t) = \sum_{s'} \bar{c}_\lambda(s',t)$:

$$\begin{aligned} \frac{\partial \log p(s|t)}{\partial \alpha_t} &= k_t \left[\frac{p_I(s|t)}{p(s|t)} - \frac{p_o(s|t)}{p(s|t)} \right] \\ \frac{\partial \log p(s|t)}{\partial \gamma} &= \frac{1-\alpha_t}{p(s|t)} \left[\frac{u(s|t)}{\bar{c}_\lambda(t)} - \frac{\bar{c}_\lambda(s,t)}{\bar{c}_\lambda(t)^2} \right] \\ \frac{\partial \log p(s|t)}{\partial \lambda_i} &= \frac{1-\alpha_t}{p(s|t)} \left[\frac{c_{\lambda'_i}(s,t)}{\bar{c}_\lambda(t)} - \frac{\bar{c}_\lambda(s,t) c_{\lambda'_i}(t)}{\bar{c}_\lambda(t)^2} \right] \end{aligned}$$

where:

$$k_t = \begin{cases} 1 & \text{fixed weight} \\ -c_I(t)/(c_I(t)+\beta)^2 & \text{MAP} \end{cases}$$

$$c_{\lambda'_i}(s,t) = f_i(s,t)(1 - w_\lambda(s,t))c_\lambda(s,t)$$

and:

$$c_{\lambda'_i}(t) = \sum_{s'} c_{\lambda'_i}(s',t).$$

3.2 Interpretation and Variants

To motivate weighting joint OUT counts as in (6), we begin with the "ideal" objective for setting multinomial phrase probabilities $\theta = \{p(s|t), \forall st\}$, which is the likelihood with respect to the true IN distribution $p_{\hat{I}}(s,t)$. Jiang and Zhai (2007) suggest the following derivation, making use of the true OUT distribution $p_{\hat{o}}(s,t)$:

$$\begin{aligned} \hat{\theta} &= \operatorname*{argmax}_{\theta} \sum_{s,t} p_{\hat{I}}(s,t) \log p_\theta(s|t) \qquad (8) \\ &= \operatorname*{argmax}_{\theta} \sum_{s,t} \frac{p_{\hat{I}}(s,t)}{p_{\hat{o}}(s,t)} p_{\hat{o}}(s,t) \log p_\theta(s|t) \\ &\approx \operatorname*{argmax}_{\theta} \sum_{s,t} \frac{p_{\hat{I}}(s,t)}{p_{\hat{o}}(s,t)} c_o(s,t) \log p_\theta(s|t), \end{aligned}$$

where $c_o(s,t)$ are the counts from OUT, as in (6). This has solutions:

$$p_{\hat{\theta}}(s|t) = \frac{p_{\hat{I}}(s,t)}{p_{\hat{o}}(s,t)} c_o(s,t) / \sum_{s'} \frac{p_{\hat{I}}(s',t)}{p_{\hat{o}}(s',t)} c_o(s',t),$$

and from the similarity to (5), assuming $\gamma = 0$, we see that $w_\lambda(s,t)$ can be interpreted as approximating $p_{\hat{I}}(s,t)/p_{\hat{o}}(s,t)$. The logistic function, whose outputs are in $[0,1]$, forces $p_{\hat{I}}(s,t) \leq p_{\hat{o}}(s,t)$. This is not unreasonable given the application to phrase pairs from OUT, but it suggests that an interesting alternative might be to use a plain log-linear weighting

[4] Note that the probabilities in (7) need only be evaluated over the support of $\tilde{p}(s,t)$, which is quite small when this distribution is derived from a dev set. Maximizing (7) is thus much faster than a typical MERT run.

function $\exp(\sum_i \lambda_i f_i(s,t))$, with outputs in $[0,\infty]$. We have not yet tried this.

An alternate approximation to (8) would be to let $w_\lambda(s,t)$ directly approximate $p_{\hat{I}}(s,t)$. With the additional assumption that (s,t) can be restricted to the support of $c_o(s,t)$, this is equivalent to a "flat" alternative to (6) in which each non-zero $c_o(s,t)$ is set to one. This variant is tested in the experiments below.

A final alternate approach would be to combine weighted joint frequencies rather than conditional estimates, ie: $c_I(s,t) + w_\lambda(s,t)c_o(,s,t)$, suitably normalized.[5] Such an approach could be simulated by a MAP-style combination in which separate $\beta(t)$ values were maintained for each t. This would make the model more powerful, but at the cost of having to learn to downweight OUT separately for each t, which we suspect would require more training data for reliable performance. We have not explored this strategy.

3.3 Simple Features

We used 22 features for the logistic weighting model, divided into two groups: one intended to reflect the degree to which a phrase pair belongs to general language, and one intended to capture similarity to the IN domain.

The 14 general-language features embody straightforward cues: frequency, "centrality" as reflected in model scores, and lack of burstiness. They are:

- total number of tokens in the phrase pair (1);
- OUT corpus frequency (1);
- OUT-corpus frequencies of rarest source and target words (2);
- perplexities for OUT IBM1 models, in both directions (2);
- average and minimum source and target word "document frequencies" in the OUT corpus, using successive 100-line pseudo-documents[6] (4); and
- average and minimum source and target word values from the OUT corpus of the following statistic, intended to reflect degree of burstiness (higher values indicate less bursty behaviour): $g/(L - L/(l+1) + \epsilon)$, where g is the sum over all sentences containing the word of the distance (number of sentences) to the nearest sentence that also contains the word, L is the total number of sentences, l is the number of sentences that contain the word, and ϵ is a small constant (4).

The 8 similarity-to-IN features are based on word frequencies and scores from various models trained on the IN corpus:

- 1gram and 2gram source and target perplexities according to the IN LM (4);[7]
- source and target OOV counts with respect to IN (2); and
- perplexities for IN IBM1 models, in both directions (2).

To avoid numerical problems, each feature was normalized by subtracting its mean and dividing by its standard deviation.

3.4 SVM Feature

In addition to using the simple features directly, we also trained an SVM classifier with these features to distinguish between IN and OUT phrase pairs. Phrase tables were extracted from the IN and OUT training corpora (not the dev as was used for instance weighting models), and phrase pairs in the intersection of the IN and OUT phrase tables were used as positive examples, with two alternate definitions of negative examples:

1. Pairs from OUT that are not in IN, but whose source phrase is.
2. Pairs from OUT that are not in IN, but whose source phrase is, and where the intersection of IN and OUT translations for that source phrase is empty.

[5] We are grateful to an anonymous reviewer for pointing this out.

[6] One of our experimental settings lacks document boundaries, and we used this approximation in both settings for consistency.

[7] In the case of the Chinese experiments below, source LMs were trained using text segmented with the LDC segmenter, as were the other Chinese models in our system.

The classifier trained using the 2nd definition had higher accuracy on a development set. We used it to score all phrase pairs in the OUT table, in order to provide a feature for the instance-weighting model.

4 Experiments

4.1 Corpora and System

We carried out translation experiments in two different settings. The first setting uses the European Medicines Agency (EMEA) corpus (Tiedemann, 2009) as IN, and the Europarl (EP) corpus (`www.statmt.org/europarl`) as OUT, for English/French translation in both directions. The dev and test sets were randomly chosen from the EMEA corpus. Figure 1 shows sample sentences from these domains, which are widely divergent.

The second setting uses the news-related subcorpora for the NIST09 MT Chinese to English evaluation[8] as IN, and the remaining NIST parallel Chinese/English corpora (UN, Hong Kong Laws, and Hong Kong Hansard) as OUT. The dev corpus was taken from the NIST05 evaluation set, augmented with some randomly-selected material reserved from the training set. The NIST06 and NIST08 evaluation sets were used for testing. (Thus the domain of the dev and test corpora matches IN.) Compared to the EMEA/EP setting, the two domains in the NIST setting are less homogeneous and more similar to each other; there is also considerably more IN text available.

The corpora for both settings are summarized in table 1.

corpus	sentence pairs
Europarl	1,328,360
EMEA train	11,770
EMEA dev	1,533
EMEA test	1,522
NIST OUT	6,677,729
NIST IN train	2,103,827
NIST IN dev	1,894
NIST06 test	1,664
NIST08 test	1,357

Table 1: Corpora

The reference medicine for Silapo is EPREX/ERYPO, which contains epoetin alfa.
Le médicament de référence de Silapo est EPREX/ERYPO, qui contient de l'époétine alfa.
—
I would also like to point out to commissioner Liikanen that it is not easy to take a matter to a national court.
Je voudrais préciser, à l'adresse du commissaire Liikanen, qu'il n'est pas aisé de recourir aux tribunaux nationaux.

Figure 1: Sentence pairs from EMEA (top) and Europarl text.

We used a standard one-pass phrase-based system (Koehn et al., 2003), with the following features: relative-frequency TM probabilities in both directions; a 4-gram LM with Kneser-Ney smoothing; word-displacement distortion model; and word count. Feature weights were set using Och's MERT algorithm (Och, 2003). The corpus was word-aligned using both HMM and IBM2 models, and the phrase table was the union of phrases extracted from these separate alignments, with a length limit of 7. It was filtered to retain the top 30 translations for each source phrase using the TM part of the current log-linear model.

4.2 Results

Table 2 shows results for both settings and all methods described in sections 2 and 3. The 1st block contains the simple baselines from section 2.1. The natural baseline (*baseline*) outperforms the pure IN system only for EMEA/EP fren. Log-linear combination (*loglin*) improves on this in all cases, and also beats the pure IN system.

The 2nd block contains the IR system, which was tuned by selecting text in multiples of the size of the EMEA training corpus, according to dev set performance. This significantly underperforms log-linear combination.

The 3rd block contains the mixture baselines. The linear LM (*lin lm*), TM (*lin tm*) and MAP TM (*map tm*) used with non-adapted counterparts perform in all cases slightly worse than the log-linear combination, which adapts both LM and TM components. However, when the linear LM is combined with a

[8] `www.itl.nist.gov/iad/mig//tests/mt/2009`

method	EMEA/EP		NIST	
	fren	enfr	nst06	nst08
in	32.77	31.98	27.65	21.65
out	20.42	17.41	19.85	15.71
baseline	33.61	31.15	26.93	21.01
loglin	35.94	32.62	28.09	21.85
ir	33.75	31.91	—	—
lin lm	35.61	31.55	28.02	21.68
lin tm	35.32	32.52	27.16	21.32
map tm	35.15	31.99	27.20	21.17
lm+lin tm	36.42	33.49	27.83	22.03
lm+map tm	36.28	33.31	28.05	22.11
iw all	36.55	33.73	28.74	22.28
iw all map	**37.01**	**33.90**	**30.04**	**23.76**
iw all flat	36.50	33.42	28.31	22.13
iw gen map	36.98	33.75	29.81	23.56
iw sim map	36.82	33.68	29.66	23.53
iw svm map	36.79	33.67	—	—

Table 2: Results, for EMEA/EP translation into English (fren) and French (enfr); and for NIST Chinese to English translation with NIST06 and NIST08 evaluation sets. Numbers are BLEU scores.

linear TM (*lm+lin tm*) or MAP TM (*lm+map TM*), the results are much better than a log-linear combination for the EMEA setting, and on a par for NIST. This is consistent with the nature of these two settings: log-linear combination, which effectively takes the intersection of IN and OUT, does relatively better on NIST, where the domains are broader and closer together. Somewhat surprisingly, there do not appear to be large systematic differences between linear and MAP combinations.

The 4th block contains instance-weighting models trained on all features, used within a MAP TM combination, and with a linear LM mixture. The *iw all map* variant uses a non-0 γ weight on a uniform prior in $p_o(s|t)$, and outperforms a version with $\gamma = 0$ (*iw all*) and the "flattened" variant described in section 3.2. Clearly, retaining the original frequencies is important for good performance, and globally smoothing the final weighted frequencies is crucial. This best instance-weighting model beats the equivalant model without instance weights by between 0.6 BLEU and 1.8 BLEU, and beats the log-linear baseline by a large margin.

The final block in table 2 shows models trained on feature subsets and on the SVM feature described in 3.4. The general-language features have a slight advantage over the similarity features, and both are better than the SVM feature.

5 Related Work

We have already mentioned the closely related work by Matsoukas et al (2009) on discriminative corpus weighting, and Jiang and Zhai (2007) on (non-discriminative) instance weighting. It is difficult to directly compare the Matsoukas et al results with ours, since our out-of-domain corpus is homogeneous; given heterogeneous training data, however, it would be trivial to include Matsoukas-style identity features in our instance-weighting model. Although these authors report better gains than ours, they are with respect to a non-adapted baseline. Finally, we note that Jiang's instance-weighting framework is broader than we have presented above, encompassing among other possibilities the use of unlabelled IN data, which is applicable to SMT settings where source-only IN corpora are available.

It is also worth pointing out a connection with Daumé's (2007) work that splits each feature into domain-specific and general copies. At first glance, this seems only peripherally related to our work, since the specific/general distinction is made for features rather than instances. However, for multinomial models like our LMs and TMs, there is a one to one correspondence between instances and features, eg the correspondence between a phrase pair (s, t) and its conditional multinomial probability $p(s|t)$. As mentioned above, it is not obvious how to apply Daumé's approach to multinomials, which do not have a mechanism for combining split features. Recent work by Finkel and Manning (2009) which re-casts Daumé's approach in a hierarchical MAP framework may be applicable to this problem.

Moving beyond directly related work, major themes in SMT adaptation include the IR (Hildebrand et al., 2005; Lü et al., 2007; Zhao et al., 2004) and mixture (Finch and Sumita, 2008; Foster and Kuhn, 2007; Koehn and Schroeder, 2007; Lü et al., 2007) approaches for LMs and TMs described above, as well as methods for exploiting monolingual in-domain text, typically by translating it automatically and then performing self training (Bertoldi and Federico, 2009; Ueffing et al., 2007; Schwenk and Senellart, 2009). There has also been some work on adapting the word alignment model prior to phrase extraction (Civera and Juan, 2007; Wu et al., 2005), and on dynamically choosing a dev set (Xu et al., 2007). Other work includes transferring latent topic distributions from source to target language for LM adaptation, (Tam et al., 2007) and adapting features at the sentence level to different categories of sentence (Finch and Sumita, 2008).

6 Conclusion

In this paper we have proposed an approach for instance-weighting phrase pairs in an out-of-domain corpus in order to improve in-domain performance. Each out-of-domain phrase pair is characterized by a set of simple features intended to reflect how useful it will be. The features are weighted within a logistic model to give an overall weight that is applied to the phrase pair's frequency prior to making MAP-smoothed relative-frequency estimates (different weights are learned for each conditioning direction). These estimates are in turn combined linearly with relative-frequency estimates from an in-domain phrase table. Mixing, smoothing, and instance-feature weights are learned at the same time using an efficient maximum-likelihood procedure that relies on only a small in-domain development corpus.

We obtained positive results using a very simple phrase-based system in two different adaptation settings: using English/French Europarl to improve a performance on a small, specialized medical domain; and using non-news portions of the NIST09 training material to improve performance on the news-related corpora. In both cases, the instance-weighting approach improved over a wide range of baselines, giving gains of over 2 BLEU points over the best non-adapted baseline, and gains of between 0.6 and 1.8 over an equivalent mixture model (with an identical training procedure but without instance weighting).

In future work we plan to try this approach with more competitive SMT systems, and to extend instance weighting to other standard SMT components such as the LM, lexical phrase weights, and lexicalized distortion. We will also directly compare with a baseline similar to the Matsoukas et al approach in order to measure the benefit from weighting phrase pairs (or ngrams) rather than full sentences. Finally, we intend to explore more sophisticated instance-weighting features for capturing the degree of generality of phrase pairs.

References

ACL. 2007. *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, Prague, Czech Republic, June.

Michel Bacchiani, Brian Roark, and Murat Saraclar. 2004. Language model adaptation with MAP estimation and the perceptron algorithm. In NAACL04 (NAA, 2004).

Nicola Bertoldi and Marcello Federico. 2009. Domain adaptation for statistical machine translation with monolingual resources. In WMT09 (WMT, 2009).

Jorge Civera and Alfons Juan. 2007. Domain adaptation in Statistical Machine Translation with mixture modelling. In WMT07 (WMT, 2007).

Hal Daumé III and Daniel Marcu. 2006. Domain Adaptation for Statistical Classifiers. *Journal of Artificial Intelligence Research*, 26:101–126.

Hal Daumé III. 2007. Frustratingly Easy Domain Adaptation. In ACL-07 (ACL, 2007).

Andrew Finch and Eiichiro Sumita. 2008. Dynamic model interpolation for statistical machine translation. In *Proceedings of the ACL Workshop on Statistical Machine Translation*, Columbus, June. WMT.

Jenny Rose Finkel and Christopher D. Manning. 2009. Hierarchical Bayesian domain adaptation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Boulder, June. NAACL.

George Foster and Roland Kuhn. 2007. Mixture-model adaptation for SMT. In WMT07 (WMT, 2007).

George Foster and Roland Kuhn. 2009. Stabilizing minimum error rate training. In WMT09 (WMT, 2009).

Almut Silja Hildebrand, Matthias Eck, Stephan Vogel, and Alex Waibel. 2005. Adaptation of the translation model for statistical machine translation based on information retrieval. In *Proceedings of the 10th EAMT Conference*, Budapest, May.

Jing Jiang and ChengXiang Zhai. 2007. Instance Weighting for Domain Adaptation in NLP. In ACL-07 (ACL, 2007).

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Proceedings of the International Conference on Acoustics,* *Speech, and Signal Processing (ICASSP) 1995*, pages 181–184, Detroit, Michigan. IEEE.

Philipp Koehn and Josh Schroeder. 2007. Experiments in domain adaptation for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 224–227, Prague, Czech Republic, June. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 127–133, Edmonton, May. NAACL.

D. C. Liu and J. Nocedal. 1989. On the limited memory method for large scale optimization. *Mathematical Programming B*, 45(3):503–528.

Yajuan Lü, Jin Huang, and Qun Liu. 2007. Improving Statistical Machine Translation Performance by Training Data Selection and Optimization. In *Proceedings of the 2007 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Prague, Czech Republic.

Spyros Matsoukas, Antti-Veikko I. Rosti, and Bing Zhang. 2009. Discriminative corpus weight estimation for machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Singapore.

NAACL. 2004. *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Boston, May.

Franz Josef Och. 2003. Minimum error rate training for statistical machine translation. In *Proceedings of the 41th Annual Meeting of the Association for Computational Linguistics (ACL)*, Sapporo, July. ACL.

Holger Schwenk and Jean Senellart. 2009. Translation model adaptation for an arabic/french news translation system by lightly-supervised training. In *Proceedings of MT Summit XII*, Ottawa, Canada, September. International Association for Machine Translation.

Yik-Cheung Tam, Ian Lane, and Tanja Schultz. 2007. Bilingual-LSA Based LM Adaptation for Spoken Language Translation. In ACL-07 (ACL, 2007).

Jorg Tiedemann. 2009. News from opus - a collection of multilingual parallel corpora with tools and interfaces. In N. Nicolov, K. Bontcheva, G. Angelova, and R. Mitkov, editors, *Recent Advances in Natural Language Processing*, volume V, pages 237–248. John Benjamins, Amsterdam/Philadelphia.

Nicola Ueffing, Gholamreza Haffari, and Anoop Sarkar. 2007. Transductive learning for statistical machine translation. In ACL-07 (ACL, 2007).

WMT. 2007. *Proceedings of the ACL Workshop on Statistical Machine Translation*, Prague, June.

WMT. 2009. *Proceedings of the 4th Workshop on Statistical Machine Translation*, Athens, March.

Hua Wu, Haifeng Wang, and Zhanyi Liu. 2005. Alignment model adaptation for domain-specific word alignment. In *Proceedings of the 43th Annual Meeting of the Association for Computational Linguistics (ACL)*, Ann Arbor, Michigan, July. ACL.

Jia Xu, Yonggang Deng, Yuqing Gao, and Hermann Ney. 2007. Domain dependent statistical machine translation. In *MT Summit XI*, Copenhagen, September.

Bing Zhao, Matthias Eck, and Stephan Vogel. 2004. Language model adaptation for statistical machine translation with structured query models. In *Proceedings of the International Conference on Computational Linguistics (COLING) 2004*, Geneva, August.

NLP on Spoken Documents without ASR

Mark Dredze, Aren Jansen, Glen Coppersmith, Ken Church
Human Language Technology Center of Excellence
Center for Language and Speech Processing
Johns Hopkins University
mdredze,aren,coppersmith,Kenneth.Church@jhu.edu

Abstract

There is considerable interest in interdisciplinary combinations of automatic speech recognition (ASR), machine learning, natural language processing, text classification and information retrieval. Many of these boxes, especially ASR, are often based on considerable linguistic resources. We would like to be able to process spoken documents with few (if any) resources. Moreover, connecting black boxes in series tends to multiply errors, especially when the key terms are out-of-vocabulary (OOV). The proposed alternative applies text processing directly to the speech without a dependency on ASR. The method finds long (~ 1 sec) repetitions in speech, and clusters them into pseudo-terms (roughly phrases). Document clustering and classification work surprisingly well on pseudo-terms; performance on a Switchboard task approaches a baseline using gold standard manual transcriptions.

1 Introduction

Can we do IR-like tasks without ASR? Information retrieval (IR) typically makes use of simple features that count terms within/across documents such as term frequency (tf) and inverse document frequency (IDF). Crucially, to compute these features, it is sufficient to count repetitions of a term. In particular, for many IR-like tasks, there is no need for an automatic speech recognition (ASR) system to label terms with phonemes and/or words.

This paper builds on Jansen et al. (2010), a method for discovering terms with zero resources. This approach identifies long, faithfully repeated patterns in the acoustic signal. These acoustic repetitions often correspond to terms useful for information retrieval tasks. Critically, this method does not require a phonetically interpretable acoustic model or knowledge of the target language.

By analyzing a large untranscribed corpus of speech, this discovery procedure identifies a vast number of repeated regions that are subsequently grouped using a simple graph-based clustering method. We call the resulting groups pseudo-terms since they typically represent a single word or phrase spoken at multiple points throughout the corpus. Each pseudo-term takes the place of a word or phrase in bag of terms vector space model of a text document, allowing us to apply standard NLP algorithms. We show that despite the fully automated and noisy method by which the pseudo-terms are created, we can still successfully apply NLP algorithms with performance approaching that achieved with the gold standard manual transcription.

Natural language processing tools can play a key role in understanding text document collections. Given a large collection of text, NLP tools can classify documents by category (classification) and organize documents into similar groups for a high level view of the collection (clustering). For example, given a collection of news articles, these tools can be applied so that the user can quickly see the topics covered in the news articles, and organize the collection to find all articles on a given topic. These tools require little or no human input (annotation) and work across languages.

Given a large collection of speech, we would like

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 460–470,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

tools that perform many of the same tasks, allowing the user to understand the contents of the collection while listening to only small portions of the audio. Previous work has applied these NLP tools to speech corpora with similar results (see Hazen et al. (2007) and the references therein.) However, unlike text, which requires little or no preprocessing, audio files are typically first transcribed into text before applying standard NLP tools. Automatic speech recognition (ASR) solutions, such as large vocabulary continuous speech recognition (LVCSR) systems, can produce an automatic transcript from speech, but they require significant development efforts and training resources, typically hundreds of hours of manually transcribed speech. Moreover, the terms that may be most distinctive in particular spoken documents often lie outside the predefined vocabulary of an off-the-shelf LVCSR system. This means that unlike with text, where many tools can be applied to new languages and domains with minimal effort, the equivalent tools for speech corpora often require a significant investment. This greatly raises the entry threshold for constructing even a minimal tool set for speech corpora analysis.

The paper proceeds as follows. After a review of related work, we describe Jansen et al. (2010), a method for finding repetitions in speech. We then explain how these repetitions are grouped into pseudo-terms. Document clustering and classification work surprisingly well on pseudo-terms; performance on a Switchboard task approaches a baseline based on gold standard manual transcriptions.

2 Related Work

In the *low resource* speech recognition regime, most approaches have focused on coupling small amounts of orthographically transcribed speech (10s of hours) with much larger collections of untranscribed speech (100s or 1000s of hours) to train accurate acoustic models with semi-supervised methods (Novotney and Schwartz, 2009). In these efforts, the goal is to reduce the annotation requirements for the construction of competent LVCSR systems. This semi-supervised paradigm was relaxed even further with the pursuit of self organizing units (SOUs), phone-like units for which acoustic models are trained with completely unsupervised methods (Garcia and Gish, 2006). Even though the move away from phonetic acoustic models improves the universality of the architecture, small amounts of orthographic transcription are still required to connect the SOUs with the lexicon.

The segmental dynamic time warping (S-DTW) algorithm (Park and Glass, 2008) was the first truly zero resource effort, designed to discover portions of the lexicon directly by searching for repeated acoustic patterns in the speech signal. This work implicitly defined a new direction for speech processing research: unsupervised spoken term discovery, the entry point of our speech corpora analysis system. Subsequent extensions of S-DTW (Jansen et al., 2010) permit applications to much larger speech collections, a flexibility that is vital to our efforts.

As mentioned above, the application of NLP methods to speech corpora have traditionally relied on high resource ASR systems to provide automatic word or phonetic transcripts. Spoken document topic classification has been an application of particular interest (Hazen et al., 2007), for which the recognized words or phone n-grams are used to characterize the documents. These efforts have produced admirable results, with ASR transcript-based performance approached that obtained using the gold standard manual transcripts. Early efforts to perform automatic topic segmentation of speech input without the aid of ASR systems have been promising (Malioutov et al., 2007), but have yet to exploit the full the range of NLP tools.

3 Identifying Matched Regions

Our goal is to identify pairs of intervals within and across utterances of several speakers that contain the same linguistic content, preferably meaningful words or terms.

The spoken term discovery algorithm of Jansen et al. (2010) efficiently searches the space of $\binom{n}{2}$ intervals, where n is the number of speech frames.[1] Jansen et al. (2010) is based on dotplots (Church and Helfman, 1993), a method borrowed from bioinformatics for finding repetitions in DNA sequences.

[1] Typically, each frame represents a 25 or 30 ms window of speech sampled every 10 ms

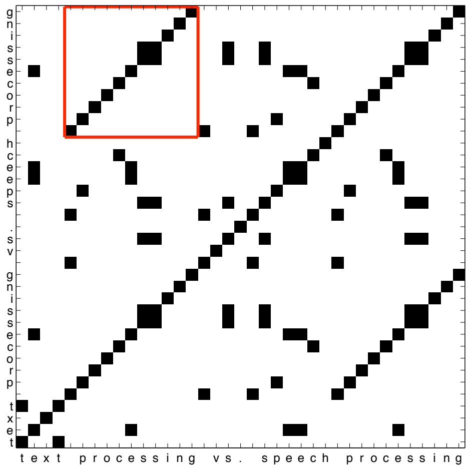


Figure 1: An example of a dotplot for the string "text processing vs. speech processing" plotted against itself. The box calls out the repeated substring: "processing."

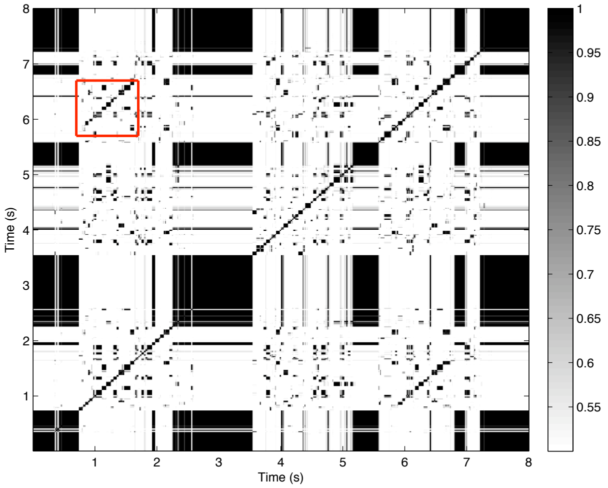


Figure 2: An example of an acoustic dotplot for 8 seconds of speech (posteriorgrams) plotted against itself. The box calls out a repetition of interest.

3.1 Acoustic Dotplots

When applied to text, the dotplot construct is remarkably simple: given character strings s_1 and s_2, the dotplot is a Boolean similarity matrix $K(s_1, s_2)$ defined as

$$K_{ij}(s_1, s_2) = \delta(s_1[i], s_2[j]).$$

Substrings common to s_1 and s_2 manifest themselves as diagonal line segments in the visualization of K. Figure 1 shows an example text dotplot where both $s1$ and $s2$ are taken to be the string "text processing vs. speech processing." The boxed diagonal line segment arises from the repeat of the word "processing," while the main diagonal line trivially arises from self-similarity. Thus, the search for line segments in K off the main diagonal provides a simple algorithmic means to identify repeated terms of possible interest, albeit sometimes partial, in a collection of text documents. The challenge is to generalize these dotplot techniques for application to speech, an inherently noisy, real-valued data stream.

The strategy is to replace character strings with frame-based speech representations of the form $\mathbf{x} = x_1, x_2, \ldots x_N$, where each $x_i \in \mathbb{R}^d$ is a d-dimensional vector space representation of the i^{th} overlapping window of the signal. Given vector time series $\mathbf{x} = x_1, x_2, \ldots x_N$ and $\mathbf{y} = y_1, y_2, \ldots y_M$ for two spoken documents, the acoustic dotplot is the real-valued $N \times M$ cosine similarity matrix $K(\mathbf{x}, \mathbf{y})$ defined as

$$K_{ij}(\mathbf{x}, \mathbf{y}) = \frac{1}{2}\left[1 + \frac{\langle x_i, y_j \rangle}{\|x_i\|\|y_j\|}\right]. \qquad (1)$$

Even though the application to speech is a distinctly noisier endeavor, sequences of frames repeated between the two audio clips will still produce approximate diagonal lines in the visualization of the matrix. The search for matched regions thus reduces to the robust search for diagonal line segments in K, which can be efficiently performed with standard image processing techniques.

Included in this procedure is the application of a diagonal median filter of duration κ seconds. The choice of κ determines an approximate threshold on the duration of the matched regions discovered. Large κ values ($\sim$ 1 sec) will produce a relatively sparse list of matches corresponding to long words or short phrases; smaller κ values ($<$ 0.5 sec) will admit shorter words and syllables that may be less informative from a document analysis perspective. Given the approximate nature of the procedure, shorter κ values also admit less reliable matches.

3.2 Posteriorgram Representation

The acoustic dotplot technique can operate on any vector time series representation of the speech signal, including a standard spectrogram. However, at the individual frame level, the cosine similarities between frequency spectra of distinct speakers producing the same phoneme are not guaranteed to be high.

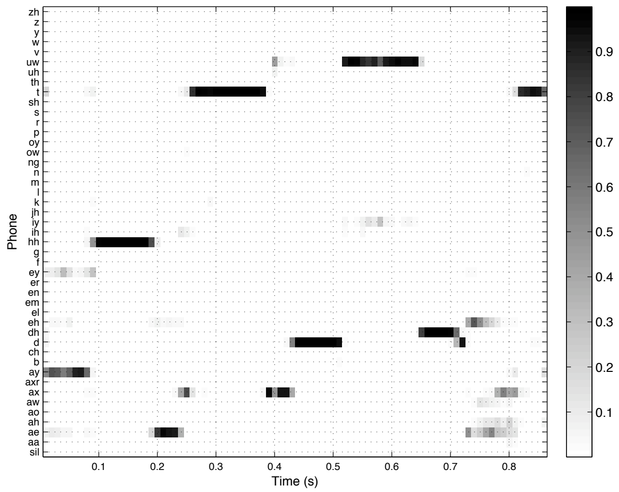


Figure 3: An example of a posteriorgram.

Thus, to perform term discovery across a multi-speaker corpus, we require a *speaker-independent* representation. Phonetic posteriorgrams are a suitable choice, as each frame is represented as the posterior probability distribution over a set of speech sounds given the speech observed at the particular point in time, which is largely speaker-independent by construction. Figure 3 shows an example posteriorgram for the utterance "I had to do that," computed with a multi-layer perceptron (MLP)-based English phonetic acoustic model (see Section 5 for details). Each row of the figure represents the posterior probability of the given phone as a function of time through the utterance and each column represents the posterior distribution over the phone set at that particular point in time.

The construction of speaker independent acoustic models typically requires a significant amount of transcribed speech. Our proposed strategy is to employ a speaker independent acoustic model trained in a high resource language or domain to interpret multi-speaker data in the zero resource target setting.[2] Indeed, we do not need to know a language to detect when a word of sufficient length has been repeated in it.[3] By computing cosine similarities of phonetic posterior distribution vectors (as opposed to reducing the speech to a one-best phonetic token sequence), the phone set used need not be matched to the target language. With this approach, a speaker-independent model trained on the phone set of a reference language may be used to perform speaker independent term discovery in any other.

In addition to speaker independence, the use of phonetic posteriorgrams introduces representational sparsity that permits efficient dotplot computation and storage. Notice that the posteriorgram displayed in Figure 3 consists of mostly near-zero values. Since cosine similarity (Equation 1) between two frames can only be high if they have significant mass on the same phone, most comparisons need not be made. Instead, we can apply a threshold and store each posteriorgram as an inverted file, performing inner product multiplies and adds only when they contribute. Using a grid of approximately 100 cores, we were able to perform the $O(n^2)$ dotplot computation and line segment search for 60+ hours of speech (corresponding to a 500 terapixel dotplot) in approximately 5 hours.

Figure 2 displays the posteriorgram dotplot for 8 seconds of speech against itself (i.e., $\mathbf{x} = \mathbf{y}$). The prominent main diagonal line results from self-similarity, and thus is ignored in the search. The boxed diagonal line segment results from two distinct occurrences of the term *one million dollars*. The large black boxes in the image result from long stretches of silence of filled pauses; fortunately, these are easily filtered with speech activity detection or simple measures of posteriorgram stability.

4 Creating Pseudo-Terms

Spoken documents will be represented as bags of pseudo-terms, where pseudo-terms are computed from acoustic repetitions described in the previous section. Let $\mathcal{M}$ be a set of matched regions (m), each consisting of a pair of speech intervals contained in the corpus ($m = [t_1^{(i)}, t_2^{(i)}], [t_1^{(j)}, t_2^{(j)}]$ indicates the speech from $t_1^{(i)}$ to $t_2^{(i)}$ is an acoustic match to the speech from $t_1^{(j)}$ to $t_2^{(j)}$). If a particular term occurs k times, the set $\mathcal{M}$ can include as many as $\binom{k}{2}$ distinct elements corresponding to that term, so we require a procedure to group them into clusters. We call the resulting clusters pseudo-terms since each

[2] A similarly-minded approach was taken in Hazen et al. (2007) and extended in Hazen and Margolis (2008), where the authors use Hungarian phonetic trigrams features to characterize English spoken documents for a topic classification task.

[3] While in this paper our acoustic model is based on our evaluation corpus, this is not a requirement of our approach. Future work will investigate performance of other acoustic models.

cluster is a placeholder for a term (word or phrase) spoken in the collection. Given the match list $\mathcal{M}$ and the pseudo-term clusters, it is relatively straightforward to represent spoken documents as bags of pseudo-terms.

To perform this pseudo-term clustering we represented matched regions as vertices in a graph with edges representing similarities between these regions. We employ a graph-clustering algorithm that extracts connected components. Let $G = (V, E)$ be an unweighted, undirected graph with vertex set V and edge set E. Each $v_i \in V$ corresponds to a single speech interval ($[t_1^{(i)}, t_2^{(i)}]$) present in $\mathcal{M}$ (each $m \in \mathcal{M}$ has a pair of such intervals, so $|V| = 2|\mathcal{M}|$) and each $e_{ij} \in E$ is an edge between vertex v_i and v_j.

The set E consists of two types of edges. The first represents repeated speech at distinct points in the corpus as determined by the match list $\mathcal{M}$. The second represents near-identical intervals in the same utterance (i.e. the same speech) since a single interval can show up in several matches in $\mathcal{M}$ and the algorithm in Section 3 explicitly ignores self-similarity. Given the intervals $[t_1^{(i)}, t_2^{(i)}]$ and $[t_1^{(j)}, t_2^{(j)}]$ contained in the same utterance and with corresponding vertices $v_i, v_j \in V$, we introduce an edge e_{ij} if fractional overlap f_{ij} exceeds some threshold τ, where $f_{ij} = \max(0, r_{ij})$ and

$$r_{ij} = \frac{(t_2^{(i)} - t_1^{(i)}) + (t_2^{(j)} - t_1^{(j)})}{\max(t_2^{(i)}, t_2^{(j)}) - \min(t_1^{(i)}, t_1^{(j)})} - 1. \quad (2)$$

From the graph G, we produce one pseudo-term for each connected component. More sophisticated edge weighting schemes would likely provide benefit. In particular, we expect improved clustering by introducing weights that reflect acoustic similarity between match intervals, rather than relying solely upon the term discovery algorithm to make a hard decision. Such confidence weights would allow even shorter pseudo-terms to be considered (by reducing κ) without greatly increasing false alarms. With such a shift, more sophisticated graph-clustering mechanisms would be warranted (e.g. Clauset et al. (2004)). We plan to pursue this in future work.

Counts	*Terms*
5	keep track of
5	once a month
2	life insurance
2	capital punishment
9	paper; newspaper
3	talking to you

Table 1: Pseudo-terms resulting from a graph clustering of matched regions ($\kappa = 0.75$, $\tau = 0.95$). Counts indicate the number of times the times the pseudo-terms appear across 360 conversation sides in development data.

Table 1 contains several examples of pseudo-terms and the matched regions included in each group. The orthographic forms are taken from the transcripts in the data (see Section 5). Note that for some pseudo-terms, the words match exactly, while for others, the phrases are distinct but phonetically similar. However, even in this case, there is often substantial overlap in the spoken terms.

5 Data

For our experiments we used the Switchboard Telephone Speech Corpus (Godfrey et al., 1992). Switchboard is a collection of roughly 2,400 two-sided telephone conversations with a single participant per side. Over 500 participants were randomly paired and prompted with a topic for discussion. Each conversation belongs to one of 70 pre-selected topics with the two sides restricted to separate channels of the audio.

To develop and evaluate our methods, we created three data sets from the Switchboard corpus: a development data set, a held out tuning data set and an evaluation data set. The development data set was created by selecting the six most commonly prompted topics (recycling, capital punishment, drug testing, family finance, job benefits, car buying) and randomly selecting 60 sides of conversations evenly across the topics (total 360 conversation sides.) This corresponds to 35.7 hours of audio. Note that each participant contributed at most one conversation side per topic, so these 360 conversation sides represent 360 distinct speakers. All algorithm development and experimentation was conducted exclusively on the development data.

For the tuning data set, we selected an additional 60 sides of conversations evenly across the same six topics used for development, for a total of 360 conversations and 37.5 hours of audio. This data was used to validate our experiments on the development data by confirming the heuristic used to select algorithmic parameters, as described below. This data was not used for algorithm development. The evaluation data set was created once parameters had been selected for a final evaluation of our methods. We selected this data by sampling 100 conversation sides from the next six most popular conversation topics (family life, news media, public education, exercise/fitness, pets, taxes), yielding 600 conversation sides containing 61.6 hours of audio.

In our experiments below, we varied the match duration κ between 0.6 s and 1.0 s and the overlap threshold τ between 0.75 and 1.0. We measured the resulting effects on the number of unique pseudo-terms generated by the process. In general, decreasing κ results in more matched regions increasing the number of pseudo-terms. Similarly, increasing τ forces fewer regions to be merged, increasing the total number of pseudo-terms. Table 2 shows how these parameters change the number of pseudo-terms (features) per document and the average number of occurrences of each pseudo-term. The user could tune these parameters to select pseudo-terms that were long and occurred in many documents. In the next sections, we consider how these parameters effect performance of various learning settings.

To provide the requisite speaker independent acoustic model, we compute English phone posteriorgrams using the multi-stream multi-layer perceptron-based architecture of Thomas et al. (2009), trained on 300 hours of conversational telephone speech. While this is admittedly a large amount of supervision, it is important to emphasize our zero resource term discovery algorithm does not rely on the phonetic interpretability of this reference acoustic model. The only requirement is that the same target language phoneme spoken by distinct speakers map to similar posterior distributions over the reference language phoneme set. Thus, even though we evaluate the system on matched-language Switchboard data, it can be just as easily applied to any target language with no language-specific knowledge or training resources required.[4]

κ	τ	*Features*	*Feat. Frequency*	*Feat./Doc.*
0.6	0.75	5,809	2.15	34.7
0.6	0.85	23,267	2.22	143.4
0.6	0.95	117,788	2.38	779.8
0.6	1.0	333,816	2.32	2153.4
0.75	0.75	8,236	2.31	52.8
0.75	0.85	18,593	2.36	121.7
0.75	0.95	48,547	2.36	318.2
0.75	1.0	90,224	2.18	546.9
0.85	0.75	5,645	2.52	39.5
0.85	0.85	8,832	2.44	59.8
0.85	0.95	15,805	2.24	98.3
0.85	1.0	24,480	2.10	142.4
1.0	0.75	1,844	2.39	12.3
1.0	0.85	2,303	2.24	14.4
1.0	0.95	3,239	2.06	18.6
1.0	1.0	4,205	1.93	22.7

Table 2: Statistics on the number of features (pseudo-terms) generated for different settings of the match duration κ and the overlap threshold τ.

6 Document Clustering

We begin by considering document clustering, a popular approach to discovering latent structure in document collections. Unsupervised clustering algorithms sort examples into groups, where each group contains documents that are similar. A user exploring a corpus can look at a few documents in each cluster to gain an overview of the content discussed in the corpus. For example, clustering methods can be used on search results to provide quick insight into the coverage of the returned documents (Zeng et al., 2004).

Typically, documents are clustered based on a bag of words representation. In the case of clustering conversations in our collection, we would normally obtain a transcript of the conversation and then extract a bag of words representation for clustering. The resulting clusters may represent topics, such as the six topics used in our switchboard data. Such groupings, available with no topic labeled training data, can be a valuable tool for understanding the contents of a speech data collection. We would like to know if similar clustering results can be obtained without the use of a manual or automatic transcript. In our case, we substitute the pseudo-terms discovered in a conversation for the transcript, representing

[4] The generalization of the speaker independence of acoustic models across languages is not well understood. Indeed, the performance of our proposed system would depend to some extent on the phonetic similarity of the target and reference language. Unsupervised learning of speaker independent acoustic models remains an important area of future research.

the document as a bag of pseudo-terms instead of actual words. Can a clustering algorithm achieve similar results along topical groups with our transcript-free representation as it can with a full transcript?

In our experiments, we use the six topic labels provided by Switchboard as the clustering labels. The goal is to cluster the data into six balanced groups according to these topics. While Switchboard topics are relatively straightforward to identify since the conversations were prompted with specific topics, we believe this task can still demonstrate the effectiveness of our representation relative to the baseline methods. After all, topic classification without ASR is still a difficult task.

6.1 Evaluation Metrics

There are numerous approaches to evaluating clustering algorithms. We consider several methods: Purity, Entropy and B-Cubed. For a full treatment of these metrics, see Amigó et al. (2009).

Purity measures the precision of each cluster, i.e., how many examples in each cluster belong to the same true topic. Purity ranges between zero and one, with one being optimal. While optimal purity can be obtained by putting each document in its own cluster, we fix the number of clusters in all experiments so purity numbers are comparable. The purity of a cluster is defined as the largest percentage of examples in a cluster that have the same topic label. Purity of the entire clustering is the average purity of each cluster:

$$\text{purity}(C, L) = \frac{1}{N} \sum_{c_i \in C} \max_{l_j \in L} |c_i \cap l_j| \qquad (3)$$

where C is the clustering, L is the reference labeling, and N are the number of examples. Following this notation, c_i is a specific cluster and l_j is a specific true label.

Entropy measures how the members of a cluster are distributed amongst the true labels. The global metric is computed by taking the weighted average of the entropy of the members of each cluster. Specifically, entropy(C, L) is given by:

$$-\sum_{c_i \in C} \frac{N_i}{N} \sum_{l_j \in L} P(c_i, l_j) \log_2 P(c_i, l_j) \qquad (4)$$

where N_i is the number of instances in cluster i, $P(c_i, l_j)$ is the probability of seeing label l_j in cluster c_i and the other variables are defined as above.

B-Cubed measures clustering effectiveness from the perspective of a user's inspecting the clustering results (Bagga and Baldwin, 1998). B-Cubed precision can be defined as an algorithm as follows: suppose a user randomly selects a single example. She then proceeds to inspect every other example that occurs in the same cluster. How many of these items will have the same true label as the selected example (precision)? B-Cubed recall operates in a similar fashion, but it measures what percentage of all examples that share the same label as the selected example will appear in the selected cluster. Since B-Cubed averages its evaluation over each document and not each cluster, it is less sensitive to small errors in large clusters as opposed to many small errors in small clusters. We include results for B-Cubed F1, the harmonic mean of precision and recall.

6.2 Clustering Algorithms

We considered several clustering algorithms: repeated bisection, globally optimal repeated bisection, and agglomerative clustering (see Karypis (2003) for implementation details). Each bisection algorithm is run 10 times and the optimal clustering is selected according to a provided criteria function (no true labels needed). For each clustering method, we evaluated several criteria functions. Additionally, we considered different scalings of the feature values (the number of times the pseudo-terms appear in each document). We found that scaling each feature by the inverse document frequency, effectively TFIDF, produced the best results, so we use that scaling in all of our experiments. We also explored various similarity metrics and found cosine similarity to be the most effective.

We used the Cluto clustering library for all clustering experiments (Karypis, 2003). In the following section, we report results for the optimal clustering configuration based on experiments on the development data.

6.3 Baselines

We compared our pseudo-term feature set performance to two baselines: (1) Phone Trigrams and

(2) Word Transcripts. The Phone Trigram baseline is derived automatically using an approach similar to Hazen et al. (2007). This baseline is based on a vanilla phone recognizer on top of the same MLP-based acoustic model (see Section 5 and the references therein for details) used to discover the pseudo-terms. In particular, the phone posteriorgrams were transformed to frame-level monophone state likelihoods (through division by the frame-level priors). These state likelihoods were then used along with frame-level phone transition probabilities to Viterbi decode each conversation side. It is important to emphasize that the reliability of phone recognizers depends on the phone set matching the application language. Using the English acoustic model in this manner on another language will significantly degrade the performance numbers reported below.

The Word Transcript baseline starts with Switchboard transcripts. This baseline serves as an upper bound of what large vocabulary recognition can provide for this task. n-gram features are computed from the transcript. Performance is reported separately for unigrams, bigrams and trigrams.

6.4 Results

To optimize parameter settings, match duration (κ) and overlap threshold (τ) were swept over a wide range ($0.6 < \kappa < 1.0$ and $0.75 < \tau < 1.0$) using a variety of clustering algorithms and training criteria. Initial results on development data showed promising performance for the default $\mathcal{I}_2$ criteria in Cluto (repeated bisection set to maximize the square root of within cluster similarity). Representative results on development data with various parameter settings for this clustering configuration appear in Table 3.

Pseudo-term Results					
κ	τ	*Features*	*Purity*	*Entropy*	B^3 F1
0.60	0.95	117,788	0.9639	0.2348	0.9306
0.60	0.96	143,299	**0.9750**	**0.1664**	**0.9518**
0.60	0.97	178,559	0.9667	0.2116	0.9366
0.60	0.98	223,511	0.9528	0.2717	0.9133
0.60	0.99	333,630	0.9583	0.2641	0.9210
0.60	1.0	333,816	0.9583	0.2641	0.9210
0.70	0.93	58,303	0.9528	0.3114	0.9105
0.70	0.94	66,054	**0.9667**	**0.2255**	**0.9358**
0.70	0.95	74,863	0.9583	0.2669	0.9210
0.70	0.96	86,070	0.9611	0.2529	0.9260
0.70	0.97	100,623	0.9639	0.2326	0.9312
0.70	0.98	117,535	0.9556	0.2821	0.9158
0.70	0.99	161,219	0.9056	0.4628	0.8372
0.70	1.0	161,412	0.9333	0.4011	0.8760

Phone Recognizer Baseline				
Type	*Features*	*Purity*	*Entropy*	B^3 F1
Phone Trigram	28,110	0.6194	1.3657	0.5256

Manual Word Transcript Baselines				
Type	*Features*	*Purity*	*Entropy*	B^3 F1
Word Unigram	7,330	**0.9917**	**0.0559**	**0.9839**
Word Bigram	74,216	0.9833	0.1111	0.9678
Word Trigram	224,934	0.9889	0.0708	0.9787

Table 3: Clustering results on development data using globally optimal repeated bisection and $\mathcal{I}_2$ criteria. The best results over the manual word transcript baselines and for each match duration (κ) are highlighted in bold. Pseudo-term results are better than the phonetic baseline and almost as good as the transcript baseline.

A few observations about results on development data. First, the three evaluation metrics are strongly correlated. Second, for each κ the same narrow range of τ values achieve good results. In general, settings of $\tau > 0.9$ were all comparable. Essentially, setting a high threshold for merging matched regions was sufficient without further tuning. Third, we observed that decreasing κ meant more features, but that these additional features did not necessarily lead to more useful features for clustering. For example, $\kappa = 0.70$ gave a small number of reasonably good features, while $\kappa = 0.60$ can give an order of magnitude more features without much of a change in clustering performance. Finally, while pseudo-term results are not as good as with the manual transcripts (unigrams), they achieve similar results. Compared with the phone trigram features determined by the phone recognizer output, the pseudo-terms perform significantly better. Note that these two automatic approaches were built using the identical MLP-based phonetic acoustic model.

We sought to select the optimal parameter settings for running on the evaluation data using the development data and the held out tuning data. We defined the following heuristic to select the optimal parameters. We choose settings for κ, τ and the clustering parameters that independently maximize the performance averaged over all runs on development data. We then selected the single run corresponding to these parameter settings and checked the result on the held out tuning data. This setting was also the best performer on the held out set, so we used these parameters for evaluation. The best performing parameters were globally optimal repeated

κ	τ	*Features*	*Purity*	*Entropy*	B^3 F1
0.70	0.98	123,901	0.9778	0.1574	0.9568
Phone Trigram		28,374	0.6389	1.2345	0.5513
Word Unigram		7,640	0.9972	0.0204	0.9945
Word Bigram		77,201	0.9972	0.0204	0.9945
Word Trigram		233,744	0.9972	0.0204	0.9945

Table 4: Results on held out tuning data. The parameters (globally optimal repeated bisection clustering with $\mathcal{I}_2$ criteria, $\kappa = 0.70$ seconds and $\tau = 0.98$) were selected using the development data and validated on tuning data. Note that the clusters produced by each manual transcript test were identical in this case.

κ	τ	*Features*	*Purity*	*Entropy*	B^3 F1
0.70	0.98	279,239	0.9517	0.3366	0.9073
Phone Trigram		31,502	0.7000	1.0496	0.6355
Word Unigram		9939	0.9883	0.0831	0.9772
Word Bigram		110,859	0.9883	0.0910	0.9771
Word Trigram		357,440	0.9900	0.0775	0.9803

Table 5: Results on evaluation data. The parameters (globally optimal repeated bisection clustering with $\mathcal{I}_2$ criteria, $\kappa = 0.7$ seconds and $\tau = 0.98$) were selected using the development data and validated on tuning data.

bisection clustering with $\mathcal{I}_2$ criteria, $\kappa = 0.7$ s and $\tau = 0.98$. Note that examining Table 3 alone may suggest other parameters, but we found our selection method to yield optimal results on the tuning data.

Results on held out tuning and evaluation data for this setting compared to the manual word transcripts and phone recognizer output are shown in Tables 4 and 5. On both the tuning data and evaluation data, we obtain similar results as on the development data. While the manual transcript baseline is better than our pseudo-term representations, the results are quite competitive. This demonstrates that useful clustering results can be obtained without a full-blown word recognizer. Notice also that the pseudo-term performance remains significantly higher than the phone recognizer baseline on both sets.

7 Supervised Document Classification

Unsupervised clustering methods are attractive since they require no human annotations. However, obtaining a few labeled examples for a simple labeling task can be done quickly, especially with crowd sourcing systems such as CrowdFlower and Amazon's Mechanical Turk (Snow et al., 2008; Callison-Burch and Dredze, 2010). In this setting, a user may listen to a few conversations and label them by topic. A supervised classification algorithm can then be trained on these labeled examples and used to automatically categorize the rest of the data. In this section, we evaluate if supervised algorithms can be trained using the pseudo-term representation of the speech.

We set up a multi-class supervised classification task, where each document is labeled using one of the six Switchboard topics. A supervised learning algorithm is trained on a sample of labeled documents and is then asked to label some test data. Results are measured in terms of accuracy. Since the documents are a balanced sample of the six topics, random guessing would yield an accuracy of 0.1667.

We proceed as with the clustering experiments. We evaluate different representations for various settings of κ and τ and different classifier parameters on the development data. We then select the optimal parameter settings and validate this selection on the held out tuning data, before generating the final representations for the evaluation once the optimal parameters have been selected.

For learning we require a multi-class classifier training algorithm. We evaluated four popular learning algorithms: a) MIRA—a large margin online learning algorithm (Crammer et al., 2006); b) Confidence Weighted (CW) learning—a probabilistic large margin online learning algorithm (Dredze et al., 2008; Crammer et al., 2009); c) Maximum Entropy—a log-linear discriminative classifier (Berger et al., 1996); and d) Support Vector Machines (SVM)—a large margin discriminator (Joachims, 1998).[5] For each experiment, we used default settings of the parameters (tuning did not significantly change the results) and 10 online iterations for the online methods (MIRA, CW). Each reported result is based on 10-fold cross validation.

Table 6 shows results for various parameter settings and the four learning algorithms on development data. As before, we observe that values for $\tau > 0.9$ tend to do well. The CW learning algorithm performs the best on this data, followed by Maximum Entropy, MIRA and SVM. The optimal κ for classification is 0.75, close to the 0.7 value selected in clustering. As before, pseudo-terms do

[5] We used the "variance" formulation with $k = 1$ for CW learning, Gaussian regularization for the Maximum Entropy classifier, and a linear kernel for the SVM.

κ	τ	*MaxEnt*	*SVM*	*CW*	*MIRA*
0.60	0.99	0.8972	0.6944	0.8667	0.8972
0.60	1.0	0.8972	0.6944	0.8639	0.8944
0.70	0.97	0.9000	0.7722	0.8500	0.8056
0.70	0.98	0.8806	0.7417	0.8917	0.8167
0.70	0.99	0.9000	0.6556	0.9194	0.9056
0.70	1.0	0.8917	0.6556	0.9194	0.9083
0.75	0.94	0.8778	0.7806	0.8639	0.8056
0.75	0.95	0.8778	0.7694	0.8889	0.8111
0.75	0.96	0.9028	0.7778	0.9000	0.8778
0.75	0.97	**0.9111**	0.7722	**0.9250**	**0.9278**
0.75	0.98	0.9056	0.7417	0.9194	0.9167
0.85	0.85	0.8639	**0.7833**	0.8500	0.8167
0.85	0.90	0.8611	0.7528	0.8611	0.8583
0.85	0.91	0.8389	0.7500	0.8722	0.8556
0.85	0.92	0.8528	0.7222	0.8944	0.8556
Phone Trigram		0.6111	0.7139	0.9138	0.5000
Word Unigram		**0.9472**	**0.8861**	0.9861	**0.9306**
Word Bigram		0.9250	0.8833	**0.9917**	0.9278
Word Trigram		0.9278	0.8611	0.9889	0.9222

Table 6: The top 15 results (measured as average accuracy across the 4 algorithms) for pseudo-terms on development data. The best pseudo-term and manual transcript results for each algorithm are bolded. All results are based on 10-fold cross validation. Pseudo-term results are better than the phonetic baseline and almost as good as the transcript baseline.

well, though not as well as the upper bound based on manual transcripts. The performance for pseudo-terms and phone trigrams are roughly comparable, though we expect pseudo-terms to be more robust across languages.

Using the same selection heuristic as in clustering, we select the optimal parameter settings, validate them on the held out tuning data, and compute results on evaluation data. The best performing configuration was for $\kappa = 0.75$ seconds and $\tau = 0.97$. Notice these parameters are very similar to the best parameters selected for clustering. Results on held out tuning and evaluation data for this setting compared to the manual transcripts are shown in Tables 7 and 8. As with clustering, we see good overall performance as compared with manual transcripts. While the performance drops, results suggest that useful output can be obtained without a transcript.

8 Conclusions

We have presented a new strategy for applying standard NLP tools to speech corpora without the aid of a large vocabulary word recognizer. Built instead on top of the unsupervised discovery of term-like units in the speech, we perform unsupervised topic clustering as well as supervised classification of spoken documents with performance approaching that achieved with the manual word transcripts, and generally matching or exceeding that achieved with a phonetic recognizer. Our study identified several opportunities and challenges in the development of NLP tools for spoken documents that rely on little or no linguistic resources such as dictionaries and training corpora.

κ	τ	*MaxEnt*	*SVM*	*CW*	*MIRA*
0.75	0.97	0.8722	0.7389	0.8972	0.8750
Phone Trigram		0.7167	0.6972	0.9056	0.5083
Word Unigram		0.9500	0.9056	0.9806	0.9250
Word Bigram		0.9444	0.9111	0.9833	0.9250
Word Trigram		0.9417	0.8972	0.9778	0.9250

Table 7: Results on held out tuning data. The parameters ($\kappa = 0.75$ seconds and $\tau = 0.97$) were selected using the development data and validated on tuning data. All results are based on 10-fold cross validation. Pseudo-term results are very close to the transcript baseline and often better than the phonetic baseline.

κ	τ	*MaxEnt*	*SVM*	*CW*	*MIRA*
0.75	0.97	0.8683	0.7167	0.7850	0.7150
Phone Trigram		0.8600	0.7750	0.9183	0.6233
Word Unigram		0.9533	0.9317	0.9850	0.9267
Word Bigram		0.9467	0.9200	0.9900	0.9367
Word Trigram		0.9383	0.9233	0.9817	0.9367

Table 8: Results on evaluation data. The parameters ($\kappa = 0.75$ seconds and $\tau = 0.97$) were selected using the development data and validated on tuning data. All results are based on 10-fold cross validation. Pseudo-term results are very close to the transcript baseline and often better than the phonetic baseline.

References

Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. 2009. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, 12(4).

A. Bagga and B. Baldwin. 1998. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 79–85. Association for Computational Linguistics.

A.L. Berger, V.J.D. Pietra, and S.A.D. Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Chris Callison-Burch and Mark Dredze. 2010. Creating speech and language data with Amazon's Mechanical

Turk. In *Workshop on Creating Speech and Language Data With Mechanical Turk at NAACL-HLT*.

K. W. Church and J. I. Helfman. 1993. Dotplot: A program for exploring self-similarity in millions of lines of text and code. *Journal of Computational and Graphical Statistics*.

Aaron Clauset, Mark E J Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. *Physical Review E*, 70.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research (JMLR)*.

Koby Crammer, Mark Dredze, and Alex Kulesza. 2009. Multi-class confidence weighted algorithms. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Mark Dredze, Koby Crammer, and Fernando Pereira. 2008. Confidence-weighted linear classification. In *International Conference on Machine Learning (ICML)*.

Alvin Garcia and Herbert Gish. 2006. Keyword spotting of arbitrary words using minimal speech resources. In *ICASSP*.

J.J. Godfrey, E.C. Holliman, and J. McDaniel. 1992. SWITCHBOARD: Telephone speech corpus for research and development. In *ICASSP*.

Timothy J. Hazen and Anna Margolis. 2008. Discriminative feature weighting using MCE training for topic identification of spoken audio recordings. In *ICASSP*.

Timothy J. Hazen, Fred Richardson, and Anna Margolis. 2007. Topic identification from audio recordings using word and phone recognition lattices. In *IEEE Workshop on Automatic Speech Recognition and Understanding*.

Aren Jansen, Kenneth Church, and Hynek Hermansky. 2010. Towards spoken term discovery at scale with zero resources. In *Interspeech*.

T. Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European Conference on Machine Learning (ECML)*.

George Karypis. 2003. CLUTO: A software package for clustering high-dimensional data sets. Technical Report 02-017, University of Minnesota, Dept. of Computer Science.

Igor Malioutov, Alex Park, Regina Barzilay, and James Glass. 2007. Making Sense of Sound: Unsupervised Topic Segmentation Over Acoustic Input. In *ACL*.

Scott Novotney and Richard Schwartz. 2009. Analysis of low-resource acoustic model self-training. In *Interspeech*.

Alex Park and James R. Glass. 2008. Unsupervised pattern discovery in speech. *IEEE Transactions of Audio, Speech, and Language Processing*.

R. Snow, B. O'Connor, D. Jurafsky, and A.Y. Ng. 2008. Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 254–263. Association for Computational Linguistics.

S. Thomas, S. Ganapathy, and H. Hermansky. 2009. Phoneme recognition using spectral envelope and modulation frequency features. In *Proc. of ICASSP*.

H.J. Zeng, Q.C. He, Z. Chen, W.Y. Ma, and J. Ma. 2004. Learning to cluster web search results. In *Conference on Research and development in information retrieval (SIGIR)*.

Fusing Eye Gaze with Speech Recognition Hypotheses to Resolve Exophoric References in Situated Dialogue

Zahar Prasov and Joyce Y. Chai
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824, USA
{prasovza,jchai}@cse.msu.edu

Abstract

In situated dialogue humans often utter linguistic expressions that refer to extralinguistic entities in the environment. Correctly resolving these references is critical yet challenging for artificial agents partly due to their limited speech recognition and language understanding capabilities. Motivated by psycholinguistic studies demonstrating a tight link between language production and human eye gaze, we have developed approaches that integrate naturally occurring human eye gaze with speech recognition hypotheses to resolve exophoric references in situated dialogue in a virtual world. In addition to incorporating eye gaze with the best recognized spoken hypothesis, we developed an algorithm to also handle multiple hypotheses modeled as word confusion networks. Our empirical results demonstrate that incorporating eye gaze with recognition hypotheses consistently outperforms the results obtained from processing recognition hypotheses alone. Incorporating eye gaze with word confusion networks further improves performance.

1 Introduction

Given a rapid growth in virtual world applications for tutoring and training, video games and simulations, and assistive technology, enabling situated dialogue in virtual worlds has become increasingly important. Situated dialogue allows human users to navigate in a spatially rich environment and carry a conversation with artificial agents to achieve specific tasks pertinent to the environment. Different from traditional telephony-based spoken dialogue systems and multimodal conversational interfaces, situated dialogue supports immersion and mobility in a visually rich environment and encourages social and collaborative language use (Byron et al., 2005; Gorniak et al., 2006). In situated dialogue, human users often need to make linguistic references, known as exophoric referring expressions (e.g., `the book to the right`), to extralinguistic entities in the environment. Reliably resolving these references is critical for dialogue success. However, reference resolution remains a challenging problem, partly due to limited speech and language processing capabilities caused by poor speech recognition (ASR), ambiguous language, and insufficient pragmatic knowledge.

To address this problem, motivated by psycholinguistic studies demonstrating a close relationship between language production and eye gaze, our previous work has incorporated naturally occurring eye gaze in reference resolution (Prasov and Chai, 2008). Our findings have shown that eye gaze can partially compensate for limited language processing and domain modeling. However, this work was conducted in a setting where users only spoke to a static visual interface. In situated dialogue, human speech and eye gaze patterns are much more complex. The dynamic nature of the environment and the complexity of spatially rich tasks have a massive influence on what the user will look at and say. It is not clear to what degree prior findings can generalize to situated dialogue. Therefore, this paper explores new studies on incorporating eye gaze for exophoric reference resolution in a fully situated virtual envi-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 471–481,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

ronment — a more realistic approximation of real world interaction. In addition to incorporating eye gaze with the best recognized spoken hypothesis, we developed an algorithm to also handle multiple hypotheses modeled as word confusion networks.

Our empirical results have demonstrated the utility of eye gaze for reference resolution in situated dialogue. Although eye gaze is much more noisy given the mobility of the user, our results have shown that incorporating eye gaze with recognition hypotheses consistently outperform the results obtained from processing recognition hypotheses alone. In addition, incorporating eye gaze with word confusion networks further improves performance. Our analysis also indicates that, although a word confusion network appears to be more complicated, the time complexity of its integration with eye gaze is well within the acceptable range for real-time applications.

2 Related Work

Prior work in reference resolution within situated dialogue has focused on using visual context to assist reference resolution during interaction. In (Kelleher and van Genabith, 2004) and (Byron et al., 2005), visual features of objects are used to model the focus of attention. This attention modeling is subsequently used to resolve references. In contrast to this line of research, here we explore the use of human eye gaze during real-time interaction to model attention and facilitate reference resolution. Eye gaze provides a richer medium for attentional information, but requires processing of a potentially noisy signal.

Eye gaze has been used to facilitate human machine conversation and automated language processing. For example, eye gaze has been studied in embodied conversational discourse as a mechanism to gather visual information, aid in thinking, or facilitate turn taking and engagement (Nakano et al., 2003; Bickmore and Cassell, 2004; Sidner et al., 2004; Morency et al., 2006; Bee et al., 2009). Recent work has explored incorporating eye gaze into automated language understanding such as automated speech recognition (Qu and Chai, 2007; Cooke and Russell, 2008), automated vocabulary acquisition (Liu et al., 2007; Qu and Chai, 2010), attention prediction (Qvarfordt and Zhai, 2005; Fang et al., 2009).

Motivated by previous psycholinguistic findings that eye gaze is tightly linked with language processing (Just and Carpenter, 1976; Tanenhous et al., 1995; Meyer and Levelt, 1998; Griffin and Bock, 2000), our prior work incorporates eye gaze into reference resolution. Our results demonstrate that such use of eye gaze can potentially compensate for a conversational systems limited language processing and domain modeling capability (Prasov and Chai, 2008). However, this work is conducted in a static visual environment and evaluated only on transcribed spoken utterances. In situated dialogue, eye gaze behavior is much more complex. Here, gaze fixations may be made for the purpose of navigation or scanning the environment rather than referring to a particular object. Referring expressions can be made to objects that are not in the user's field of view, but were previously visible on the interface. Additionally, users may make egocentric spatial references (e.g. "the chair on the left") which require contextual knowledge (e.g. the users position in the environment) in order to resolve. Therefore, the focus of our work here is on exploring these complex user behaviors in situated dialogue and examining how to combine eye gaze with ASR hypotheses for improved reference resolution.

Alternative ASR hypotheses have been used in many different ways in speech driven systems. Particularly, in (Mangu et al., 2000) multiple lattice alignment is used for construction of word confusion networks and in (Hakkani-Tür et al., 2006) word confusion networks are used for named entity detection. In the study presented here, we apply word confusion networks (to represent ASR hypotheses) along with eye gaze to the problem of reference resolution.

3 Data Collection

In this investigation, we created a 3D virtual world (using the Irrlicht game engine[1]) to support situated dialogue. We conducted a Wizard of Oz study in which the user must collaborate with a remote artificial agent cohort (controlled by a human) to solve a treasure hunting task. The cohort is an "expert" in treasure hunting and has some knowledge regard-

[1] http://irrlicht.sourceforge.net/

ing the locations of the treasure items, but cannot see the virtual environment. The user, immersed in the virtual world, must navigate the environment and conduct a mixed-initiative dialogue with the agent to find the hidden treasures. During the experiments, a noise-canceling microphone was used to record user speech and the Tobii 1750 display-mounted eye tracker was used to record user eye movements.

A snapshot of user interaction with the treasure hunting environment is shown in Figure 1. Here, the user's eye fixation is represented by the white dot and saccades (eye movements) are represented by white lines. The virtual world contains 10 rooms with a total of 155 unique objects that encompass 74 different object types (e.g. chair or plant).

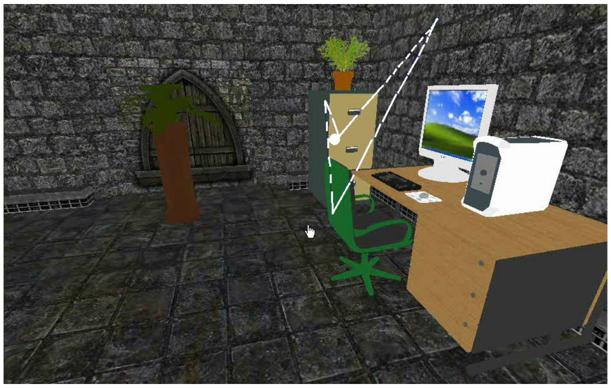


Figure 1: Snapshot of the situated treasure hunting environment

Table 1 shows a portion of a sample dialogue between a user and the expert. Each S_i represents a system utterance and each U_i represents a user utterance. We focus on resolving exophoric referring expressions, which are enclosed in brackets here. In our dataset, an exophoric referring expression is a non-pronominal noun phrase that refers to an entity in the extralinguistic environment. It may be an evoking reference that initially refers to a new object in the virtual world (e.g. `an axe` in utterance U_2) or a subsequent reference to an entity in the virtual world which has previously been mentioned in the dialogue (e.g. `an axe` in utterance U_3). In our study we focus on resolving exophoric referring expressions because they are tightly coupled with a user's eye gaze behavior.

From this study, we constructed a parallel spoken utterance and eye gaze corpus. Utterances, which are separated by a long pause (500 ms) in speech, are automatically recognized using the Microsoft Speech SDK. Gaze fixations are characterized by objects in the virtual world that are fixated via a user's eye gaze. When a fixation points to multiple spatially overlapping objects, only the one in the forefront is deemed to be fixated. The data corpus was transcribed and annotated with 2204 exophoric referring expressions amongst 2052 utterances from 15 users.

S_1	Describe what you're doing.
U_1	I just came out from the room that I started and i see **[one long sword]**
U_2	**[one short sword]** and **[an axe]**
S_2	Compare these objects.
U_3	one of them is long and one of them is really short, and i see **[an axe]**

Table 1: A conversational fragment demonstrating interaction with exophoric referring expressions.

	Utterance: i just came out from the room that i started and **i see [one long sword]**
H_t :	. . . i_5210 see_5410 [one_5630 long_6080 sword_6460]
H_1:	. . . icy_5210 winds_5630 along_6080 so_6460 words_68000
H_2:	. . . icy_5210 [wine_5630] along_6080 so_6460 words_6800
. . .	
H_{25}:	. . . icy_5210 winds_5630 [long_6080 sword_6460]
. . .	

Table 2: Sample n-best list of recognition hypotheses

4 Word Confusion Networks

For each user utterance in our dataset, an n-best list (with $n = 100$) of recognition hypotheses ranked in order of likelihood is produced by the Microsoft Speech SDK. One way to use the speech recognition results (as in most speech applications) is to use the top ranked recognition hypothesis. This may not be the best solution because a large amount of information is being ignored. Table 2 demonstrates this problem. Here, the number after the underscore denotes a timestamp associated with each recognized spoken word. The strings enclosed in brackets denote recognized referring expressions. In this example, the manual transcription of the original utterance is shown by H_t. In this case, the system must first identify `one long sword` as a referring expression and then resolve it to the correct set of entities in the virtual world. However, not until the twenty fifth ranked recognition hypothesis H_{25}, do we see a referring expression closest to the actual uttered referring expression. Moreover, in utterances with multiple referring expressions, there may not be a single recognition hypothesis that contains all referring expressions, but each referring expression may be contained in some recognition hypothesis. Thus, it is desirable to consider the entire n-best list of hypotheses.

To address this issue, we adopted the word confusion network (WCN): a compact representation of a word lattice or n-best list (Mangu et al., 2000). A WCN captures alternative word hypotheses and their corresponding posterior probabilities in time-ordered sets. In addition to being compact, an important feature for efficient post-processing of recognition hypotheses for real-time systems, WCNs are capable of representing more competing hypotheses than either n-best lists or word lattices. Figure 2 shows an example of a WCN for the utterance "...I see one long sword" along with a timeline (in milliseconds) depicting the eye gaze fixations to potential referent objects that correspond to the utterance. The confusion network shows competing word hypotheses along with corresponding probabilities in log scale.

Using our data set, we can show that word confusion networks contain significantly more words that can compose a referring expression than the top recognition hypothesis. The confusion network keyword error rate (KWER) is 0.192 compared to a 1-best list KWER of 0.318, where a keyword is a word that can be contained in a referring expression. The overall WER for word confusion networks and 1-best lists are 0.315 and 0.460, respectively. The reported WCN word error rates are all oracle word error rates reflecting the best WER that can be attained using any path in the confusion network. One more important feature of word confusion networks is that they provide time alignment for words that occur at approximately the same time interval in competing hypotheses. This is not only useful for efficient syntactic parsing, which is necessary for identifying referring expressions, but also critical for integration with time aligned gaze streams.

5 Reference Resolution Algorithm

We have developed an algorithm that combines an n-best list of speech recognition hypotheses with dialogue, domain, and eye-gaze information to resolve exophoric referring expressions. There are three inputs to the multimodal reference resolution algorithm for each utterance: (1) an n-best list of alternative speech recognition hypotheses ($n = 100$ for a WCN and $n = 1$ for the top recognized hypothesis), (2) a list of fixated objects (by eye gaze) that temporally correspond to the spoken utterance and (3) a set of potential referent objects. Since during the treasure hunting task people typically only speak about objects that are visible or have recently been visible on the screen, an object is considered to be a potential referent if it is present within a close proximity (in the same room) of the user while an utterance is spoken.

The multimodal reference resolution algorithm proceeds with the following four steps:

Step 1: construct word confusion network A word confusion network is constructed out of the input n-best list of alternative recognition hypotheses with the SRI Language Modeling (SRILM) toolkit (Stolcke, 2002) using the procedure described in (Mangu et al., 2000). This procedure aligns words from the n-best list into equivalence classes. First, instances of the same word containing approximately the same starting and ending timestamps are clustered. Then, equivalence classes with common time ranges are merged. For each competing word hypothesis its probability is computed by summing the posteriors of all utterance hypotheses containing this word. In our work, instead of using the actual posterior probability of each utterance hypothesis (which was not available), we assigned each utterance hypothesis a probability based on its position in the ranked list. Figure 2 depicts a portion of the resulting word confusion network (showing competing word hypotheses and their probabilities in log scale) constructed from the n-best list in Table 2.

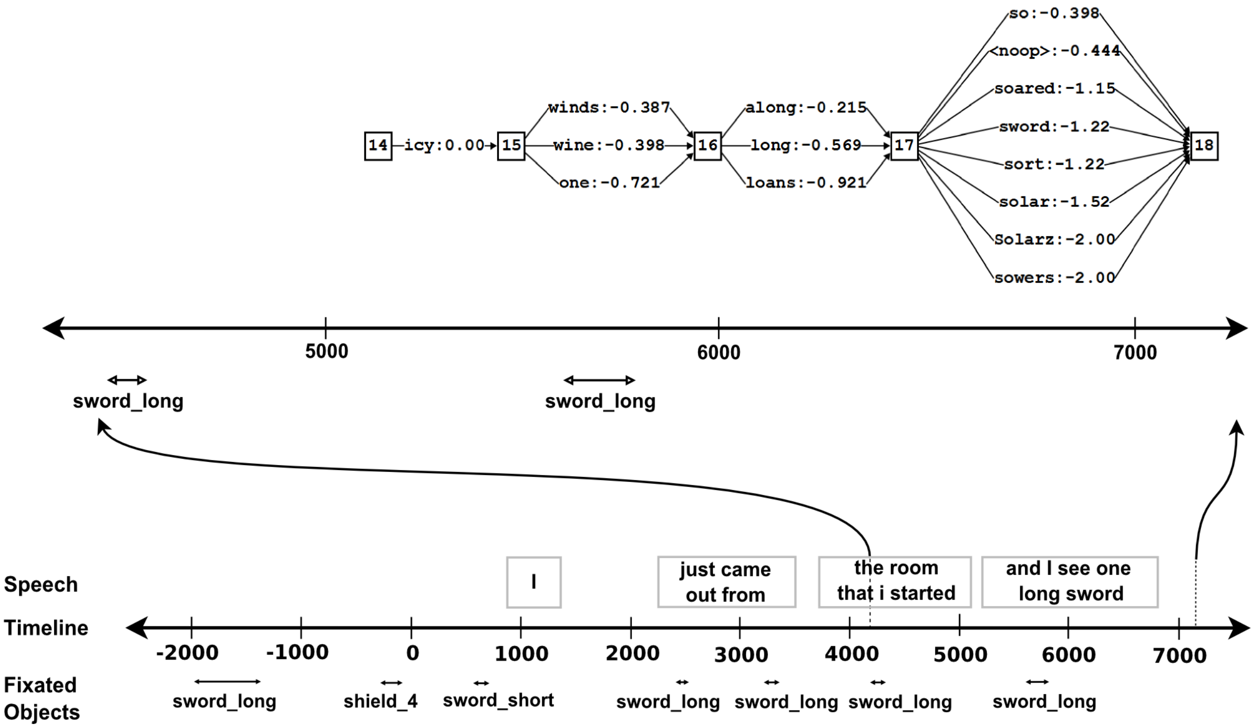


Figure 2: Sample parallel speech and eye gaze data streams, including a portion of the sample WCN

Step 2: extract referring expressions from WCN The word confusion network is syntactically parsed using a modified version of the CYK (Cooke and Schwartz, 1970; Kasami, 1965; Younger, 1967) parsing algorithm that is capable of taking a word confusion network as input rather than a single string. We call this the CYK-WCN algorithm. To do the parsing, we applied a set of grammar rules largely derived from a different domain in our previous work (Prasov and Chai, 2008). A parse chart of the sample word confusion network is shown in Table 3. Here, just as in the CYK algorithm the chart is filled in from left to right then bottom to top. The difference is that the chart has an added dimension for competing word hypotheses. This is demonstrated in position 15 of the WCN, where `one` and `wine` are two nouns that constitute competing words. Note that some words from the confusion network are not in the chart (e.g. `winds`) because they are out of vocabulary. The result of the syntactic parsing is that the parts of speech of all sub-phrases in the confusion network are identified. Next, a set of all exophoric referring expressions (i.e. non-pronominal noun phrases) found in the word confusion network are extracted. Each referring expression has a corresponding confidence score, which can be computed in many many different ways. Currently, we simply take the mean of the probability scores of the expression's constituent words. The sample WCN has four such phrases (shown in bold in Table 3): `wine` at position 15 with length 1, `one long sword` at position 15 with length 3, `long sword` at position 16 with length 2, and `sword` at position 17 with length 1.

Step 3: resolve referring expressions Each referring expression r_j is resolved to the top k potential referent objects according to the probability $P(o_i|r_j)$, where k is determined by information from the linguistic expressions. $P(o_i|r_j)$ is determined using the following expression:

$$P(o_i|r_j) = \frac{AS(o_i)^{\alpha} \times Compat(o_i, r_j)^{1-\alpha}}{\sum_i AS(o_i)^{\alpha} \times Compat(o_i, r_j)^{1-\alpha}} \tag{1}$$

In this equation,

- AS: Attentional salience score of a particu-

length	...						
	5						
	4						
	3		**NP → NUM Adj-NP**				
	2			Adj-NP → ADJ N, **NP → Adj-NP**			
	1		(1) N → wine, **NP → N** (2) NUM → one	ADJ → long	N → sword, **NP → N**		
	...	14	15	16	17	18	...
	WCN position						

Table 3: Syntactic parsing of word confusion network

lar object o_i, which is determined based on the gaze *fixation intensity* of an object at the start time of referring expression r_j. The fixation intensity of an object is defined as the amount of time that the object is fixated during a predefined time window W (Prasov and Chai, 2008). As in (Prasov and Chai, 2008), we set $W = [-1500..0]$ ms relative to the beginning of referring expression r_j.

- $Compat$: Compatibility score, which specifies whether the object o_i is compatible with the information specified by the referring expression r_j. Currently, the compatibility score is set to 1 if referring expression r_j and object o_i have the same object type (e.g. chair), and 0 otherwise.

- α: Importance weight, in the range $[0..1]$, of attentional salience relative to compatibility. A high α value indicates that the attentional salience score based on eye gaze carries more weight in deciding referents, while a low α value indicates that compatibility carries more weight. In this work, we set $\alpha = 0.5$ to indicate equal weighting between attentional salience and compatibility. If we do not want to integrate eye gaze in reference resolution, we can set $\alpha = 0.0$. In this case, reference resolution will be purely based on compatibility between visual objects and information specified via linguistic expressions.

Once all probabilities are calculated, each referring expression is resolved to a set of referent objects. Finally, this results in a set of *(referring expression, referent object set)* pairs with confidence scores, which are determined by two components. The first component is the confidence score of the referring expression, which is explained in the Step 1 of the algorithm. The second component is the probability that the referent object set is indeed the referent of this expression (which is determined by Equation 1). There are various ways to combine these two components together to form an overall confidence score for the pair. Here we simply multiply the two components. The confidence score for the pair is used in the following step to prune unlikely referring expressions.

Step 4: post-prune The resulting set of *(referring expression, referent object set)* pairs is pruned to remove pairs that fall under one of the following two conditions: (1) the pair has a confidence score equal to or below a predefined threshold ϵ (currently, the threshold is set to 0 and thus keeps all resolved pairs) and (2) the pair temporally overlaps with a higher confidence pair. For example, in Table 3, the referring expressions `one long sword` and `wine` overlap in position 15. Finally, the resulting *(referring expression, referent object set)* pairs are sorted in ascending order according to their constituent referring expression timestamps.

6 Experimental Results

Using our data, described in Section 3, we applied the multimodal reference resolution algorithm described in Section 5. All of the data is used to report the experimental results. Reference resolution model parameters are set based on our prior work in a different domain (Prasov and Chai, 2008). For each utterance we compare the reference resolution performance with and without the integration of eye gaze information. We also evaluate using a word confusion network compared to a 1-best list to model speech recognition hypotheses. For perspective, reference resolution with recognized speech input is compared with transcribed speech.

6.1 Evaluation Metrics

The reference resolution algorithm outputs a list of *(referring expression, referent object set)* pairs for each utterance. We evaluate the algorithm by comparing the generated pairs to the annotated "gold standard" pairs using F-measure. We perform the following two types of evaluation:

- Lenient Evaluation: Due to speech recognition errors, there are many cases in which the algorithm may not return a referring expression that exactly matches the gold standard referring expression. It may only match based on the object type. For example, the expressions `one long sword` and `sword` are different, but they match in terms of the intended object type. For applications in which it is critical to identify the objects referred to by the user, precisely identifying uttered referring expressions may be unnecessary. Thus, we evaluate the reference resolution algorithm with a lenient comparison of *(referring expression, referent object set)* pairs. In this case, two pairs are considered a match if at least the object types specified via the referring expressions match each other and the referent object sets are identical.

- Strict Evaluation: For some applications it may be important to identify exact referring expressions in addition to the objects they refer to. This is important for applications that attempt to learn a relationship between referring expressions and referenced objects. For example, in automated vocabulary acquisition, words other than object types must be identified so the system can learn to associate these words with referenced objects. Similarly, in systems that apply priming for language generation, identification of the exact referring expressions from human users could be important. Thus, we also evaluate the reference resolution algorithm with a strict comparison of *(referring expression, referent object set)* pairs. In this case, a referring expression from the system output needs to exactly match the corresponding expression from the gold standard.

6.2 Role of Eye Gaze

We evaluate the effect of incorporating eye gaze information into the reference resolution algorithm using the top best recognition hypothesis (*1-best*), the word confusion network (*WCN*), and the manual speech transcription (*Transcription*). Speech transcription, which contains no recognition errors, demonstrates the upper bound performance of our approach. When no gaze information is used, reference resolution solely depends on linguistic and semantic processing of referring expressions. Table 4 shows the lenient reference resolution evaluation using F-measure. This table demonstrates that lenient reference resolution is improved by incorporating eye gaze information. This effect is statistically significant in the case of transcription and 1-best ($p < 0.0001$ and $p < 0.009$, respectively) and marginal ($p < 0.07$) in the case of WCN.

Configuration	Without Gaze	With Gaze
Transcription	0.619	0.676
WCN	0.524	0.552
1-best	0.471	0.514

Table 4: Lenient F-measure Evaluation

Configuration	Without Gaze	With Gaze
Transcription	0.584	0.627
WCN	0.309	0.333
1-best	0.039	0.035

Table 5: Strict F-measure Evaluation

Table 5 shows the strict reference resolution evaluation using F-measure. As can be seen in the table, incorporating eye gaze information significantly ($p < 0.0024$) improves reference resolution performance when using transcription and marginally ($p < 0.113$) in the case of WCN optimized for strict evaluation. However there is no difference for the 1-best hypotheses which result in extremely low performance. This observation is not surprising since 1-best hypotheses are quite error prone and less likely to produce the exact expressions.

Since eye gaze can be used to direct navigation in a mobile environment as in situated dialogue, there could be situations where eye gaze does not reflect the content of the corresponding speech. In such situations, integrating eye gaze in reference resolution could be detrimental. To further understand the role of eye gaze in reference resolution, we applied our reference resolution algorithm only to utterances where speech and eye gaze are considered closely coupled (i.e., eye gaze reflects the content of speech). More specifically, following the previous work (Qu and Chai, 2010), we define a *closely coupled* utterance as one in which at least one noun or adjective describes an object that has been fixated by the corresponding gaze stream.

Table 6 and Table 7 show the performance based on closely coupled utterances using lenient and strict evaluation, respectively. In the lenient evaluation, reference resolution performance is significantly improved for all input configurations when eye gaze information is incorporated ($p < 0.0001$ for transcription, $p < 0.015$ for WCN, and $p < 0.0022$ for 1-best). In each case the closely coupled utterances achieve higher performance than the entire set of utterances evaluated in Table 5. Aside from the 1-best case, the same is true when using strict evaluation ($p < 0.0006$ for transcription and $p < 0.046$ for WCN optimized for strict evaluation). This observation indicates that in situated dialogue, some mechanism to predict whether a gaze stream is closely coupled with the corresponding speech content can be beneficial in further improving reference resolution performance.

Configuration	Without Gaze	With Gaze
Transcription	0.616	0.700
WCN	0.523	0.570
1-best	0.473	0.537

Table 6: Lenient F-measure Evaluation for Closely Coupled Utterances

6.3 Role of Word Confusion Network

The effect of incorporating eye gaze with WCNs rather than 1-best recognition hypotheses into reference resolution can also be seen in Tables 4 and 5. Table 4 shows a significant improvement when using WCNs rather than 1-best hypotheses for both with ($p < 0.015$) and without ($p < 0.0012$) eye gaze configurations. Similarly, Table 5 shows a significant improvement in strict evaluation when using WCNs rather than 1-best hypotheses for both with ($p < 0.0001$) and without ($p < 0.0001$) eye gaze configurations. These results indicate that using word confusion networks improves both lenient and strict reference resolution. This observation is not surprising since identifying correct linguistic expressions will enable better search for semantically matching referent objects.

Configuration	Without Gaze	With Gaze
Transcription	0.579	0.644
WCN	0.307	0.345
1-best	0.045	0.038

Table 7: Strict F-measure Evaluation for Closely Coupled Utterances

Although WCNs lead to better performance, utilizing WCNs is more computationally expensive compared to 1-best recognition hypotheses. Nevertheless, in practice, WCN depth, which specifies the maximum number of competing word hypotheses in any position of the word confusion network, can be limited to a certain value $|d|$. For example, in Figure 2 the depth of the shown WCN is 8 (there are 8 competing word hypotheses in position 17 of the WCN). The WCN depth can be limited by pruning word alternatives with low probabilities until, at most, the top $|d|$ words remain in each position of the WCN. It is interesting to observe how limiting WCN depth can affect reference resolution performance. Figure 3 demonstrates this observation. In this figure the resolution performance (in terms of lenient evaluation) for WCNs of varying depth is shown as dashed lines for with and without eye gaze configurations. As a reference point, the performance when utilizing 1-best recognition hypotheses is shown as solid lines. It can be seen that as the depth increases, the performance also increases until the depth reaches 8. After that, there is no performance improvement.

7 Discussion

In Section 6.2 we have shown that incorporating eye gaze information improves reference resolution performance. Eye-gaze information is particu-

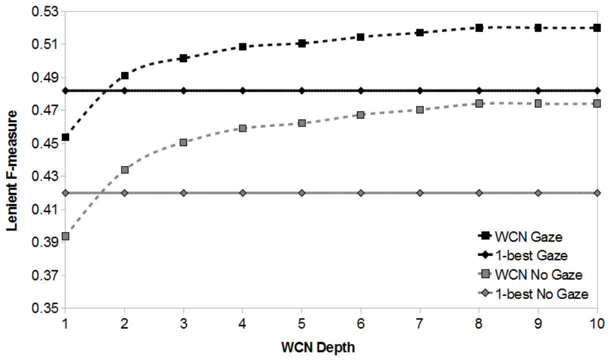


Figure 3: Lenient F-measure at each WCN Depth

larly helpful for resolving referring expressions that are ambiguous from the perspective of the artificial agent. Consider a scenario where the user utters a referring expression that has an equivalent semantic compatibility with multiple potential referent objects. For example, in a room with multiple books, the user utters "the open book to the right", but only the phrase "the book" is recognized by the ASR. If a particular book is fixated during interaction, there is a high probability that it is indeed being referred to by the user. Without eye gaze information, the semantic compatibility alone could be insufficient to resolve this referring expression. Thus, when eye gaze information is incorporated, the main source of performance improvement comes from better identification of potential referent objects.

In Section 6.3 we have shown that incorporating multiple speech recognition hypotheses in the form of a word confusion network further improves reference resolution performance. This is especially true when exact referring expression identification is required (F-measure of 0.309 from WCNs compared to F-measure of 0.039 from 1-best hypotheses). Using a WCN improves identification of low-probability referring expressions. Consider a scenario where the top recognition hypothesis of an utterance contains no referring expressions or an incorrect referring expression that has no semantically compatible potential referent objects. If a referring expression with a high compatibility value to some potential referent object is present in a lower probability hypothesis, this referring expression can only be identified when a WCN rather than a 1-best hypothesis is utilized. Thus, when word confusion networks are incorporated, the main source of performance improvement comes from better referring expression identification.

7.1 Computational Complexity

One potential concern of using word confusion networks rather than 1-best hypotheses is that they are more computationally expensive to process. The asymptotic computational complexity for resolving the referring expressions using the algorithm presented in this work with a WCN is the summation of three components: (1) $O(|G| \cdot |d|^2 \cdot |w|^3)$ for confusion network construction and parsing, (2) $O(|r| \cdot |O| \cdot log(|O|))$ for reference resolution, and (3) $O(|r|^2)$ for selection of *(referring expression, referent object set)* pairs. Here, $|w|$ is the number of words in the input speech signal (or, more precisely, the number of words in the longest ASR hypothesis for a given utterance); $|G|$ is the size of the parsing grammar; $|d|$ is the depth of the constructed word confusion network; $|O|$ is the number of potential referent objects for each utterance; and $|r|$ is the number of referring expressions that are extracted from the word confusion network.

The complexity is dominated by the word confusion network construction and parsing. Also, both the number of words in an input utterance ASR hypothesis $|w|$ and the number of referring expressions in a word confusion network $|r|$ are dependent on utterance length. In our study, interactive dialogue is encouraged and, thus, utterances are typically short; with a mean length of 6.41 words and standard deviation of 4.35 words. The longest utterances in our data set has 31 words. WCN depth $|d|$ has a mean of 10.1, a standard deviation of 8.1, and a maximum 89 words. In practice, as shown in Section 6.3, limiting $|d|$ to 8 words achieves comparable reference resolution results as using a full word confusion network.

To demonstrate the applicability of our reference resolution algorithm for real-time processing, we applied it on the data corpus presented in Section 3. This corpus contains utterances with a mean input time of 2927.5 ms and standard deviation of 1903.8 ms. On a 2.4 GHz AMD Athlon(tm) 64 X2 Dual Core Processor, the runtimes resulted in a real time factor of 0.0153 on average. Thus, on average, an utterance from this corpus can be processed in just under 45 ms, which is well within the range of acceptable real-time performance.

7.2 Error Analysis

As can be seen in Section 6, even when using transcribed data, reference resolution performance still has room for improvement (achieving the highest lenient F-measure of 0.700 when eye gaze is utilized for resolving closely coupled utterances). In this section, we elaborate on the potential error sources. Specifically, we discuss two types of error: (1) a referring expression is incorrectly recognized or (2) a recognized referring expression is not resolved to a correct referent object set.

Given transcribed data, which simulates perfectly recognized utterances, all referring expression recognition errors arise due to incorrect language processing. Most of these errors occur because an incorrect part of speech (POS) tag is assigned to a word, or an out-of-vocabulary (OOV) word is encountered, or the parsing grammar has insufficient coverage. A particularly interesting parsing problem occurs due to the nature of spoken language. Since punctuation is sometimes unavailable, given an utterance with several consecutive nouns, it is unclear which of these nouns should be treated as head nouns and which should be treated as noun modifiers. For example, in the utterance "there is *a desk lamp* table and two chairs" it is unclear if the italicized expression should be parsed as a single phrase or as a list of (two) phrases `a desk` and `lamp`. Thus, some timing information should be used for disambiguation.

Object set identification errors are more prevalent than referring expression recognition errors. The majority of these errors occur because a referring expression is ambiguous from the perspective of the conversational system and there is not enough information to choose amongst multiple potential referent objects due to limited speech recognition and domain modeling. One reason for this is that a referring expression may be resolved to an incorrect number of referent objects. Another reason is that a pertinent object attribute or a distinguishing spatial relationship between objects specified by the user cannot be established by the system. For example, during the utterance "I see *a vase* left of the table" there are two vases visible on the screen creating an ambiguity if the phrase `left of` is not processed correctly. This is caused by an inadequate representation of spatial relationships and processing of spatial language. One more reason for potential ambiguity is the lack of pragmatic knowledge that can support adequate inference. For example, when the user refers to two *sofa* objects using the phrase "*an armchair* and *a sofa*", the system lacks pragmatic knowledge to indicate that `arm chair` refers to the smaller of the two objects. Some of these errors can be avoided when eye gaze information is available to the system. However, due to the noisy nature of eye gaze data, many such referring expressions remain ambiguous even when eye gaze information is considered.

8 Conclusion

In this work, we have examined the utility of eye gaze and word confusion networks for reference resolution in situated dialogue within a virtual world. Our empirical results indicate that incorporating eye gaze information with recognition hypotheses is beneficial for the reference resolution task compared to only using recognition hypotheses. Furthermore, using a word confusion network rather than the top best recognition hypothesis further improves reference resolution performance. Our findings also demonstrate that the processing speed necessary to integrate word confusion networks with eye gaze information is well within the acceptable range for real-time applications.

Acknowledgments

This work was supported by IIS-0347548 and IIS-0535112 from the National Science Foundation. We would like to thank anonymous reviewers for their valuable comments and suggestions.

References

N. Bee, E. André, and S. Tober. 2009. Breaking the ice in human-agent communication: Eye-gaze based initiation of contact with an embodied conversational agent. In *Proceedings of the 9th International Conference on Intelligent Virtual Agents (IVA'09)*, pages 229–242. Springer.

T. Bickmore and J. Cassell, 2004. *Social Dialogue with Embodied Conversational Agents*, chapter Natural, Intelligent and Effective Interaction with Multimodal Dialogue Systems. Kluwer Academic.

D. K. Byron, T. Mampilly, and T. Sharma, V.and Xu. 2005. Utilizing visual attention for cross-modal coreference interpretation. In *Spring Lecture Notes in Computer Science: Proceedings of CONTEXT-05*, pages 83–96.

N. J. Cooke and M. Russell. 2008. Gaze-contingent automatic speech recognition. *IET Signal Processing*, 2(4):369–380, December.

J. Cooke and J. T. Schwartz. 1970. Programming languages and their compilers: Preliminary notes. Technical report, Courant Institute of Mathematical Science.

R. Fang, J. Y. Chai, and F. Ferreira. 2009. Between linguistic attention and gaze fixations in multimodal conversational interfaces. In *The 11th International Conference on Multimodal Interfaces (ICMI)*.

P. Gorniak, J. Orkin, and D. Roy. 2006. Speech, space and purpose: Situated language understanding in computer games. In *Twenty-eighth Annual Meeting of the Cognitive Science Society Workshop on Computer Games*.

Z. M. Griffin and K. Bock. 2000. What the eyes say about speaking. In *Psychological Science*, volume 11, pages 274–279.

D. Hakkani-Tür, F. Béchet, G. Riccardi, and G. Tur. 2006. Beyond asr 1-best: Using word confusion networks in spoken language understanding. *Computer Speech and Language*, 20(4):495–514.

M. A. Just and P. A. Carpenter. 1976. Eye fixations and cognitive processes. In *Cognitive Psychology*, volume 8, pages 441–480.

T. Kasami. 1965. An efficient recognition and syntax-analysis algorithm for context-free languages. Scientific report AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedford, Massachusetts.

J. Kelleher and J. van Genabith. 2004. Visual salience and reference resolution in simulated 3-d environments. *Artificial Intelligence Review*, 21(3).

Y. Liu, J. Y. Chai, and R. Jin. 2007. Automated vocabulary acquisition and interpretation in multimodal conversational systems. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL)*.

L. Mangu, E. Brill, and A. Stolcke. 2000. Finding consensus in speech recognition: word error minimization and other applications of confusion networks. *Computer Speech and Language*, 14(4):373–400.

A. S. Meyer and W. J. M. Levelt. 1998. Viewing and naming objects: Eye movements during noun phrase production. In *Cognition*, volume 66, pages B25–B33.

L.-P. Morency, C. M. Christoudias, and T. Darrell. 2006. Recognizing gaze aversion gestures in embodied conversational discourse. In *International Conference on Multimodal Interfaces (ICMI)*.

Y. I. Nakano, G. Reinstein, T. Stocky, and J. Cassell. 2003. Towards a model of face-to-face grounding. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL'03)*, pages 553–561.

Z. Prasov and J. Y. Chai. 2008. What's in a gaze? the role of eye-gaze in reference resolution in multimodal conversational interfaces. In *Proceedings of 13th International Conference on Intelligent User interfaces (IUI)*, pages 20–29.

S. Qu and J. Y. Chai. 2007. An exploration of eye gaze in spoken language processing for multimodal conversational interfaces. In *Proceedings of the Conference of the North America Chapter of the Association of Computational Linguistics (NAACL)*.

S. Qu and J. Y. Chai. 2010. Context-based word acquisition for situated dialogue in a virtual world. *Journal of Artificial Intelligence Research*, 37:347–377, March.

P. Qvarfordt and S. Zhai. 2005. Conversing with the user based on eye-gaze patterns. In *Proceedings Of the Conference on Human Factors in Computing Systems*. ACM.

C. L. Sidner, C. D. Kidd, C. Lee, and N. Lesh. 2004. Where to look: A study of human-robot engagement. In *Proceedings of the 9th international conference on Intelligent User Interfaces (IUI'04)*, pages 78–84. ACM Press.

A. Stolcke. 2002. SRILM an extensible language modeling toolkit, confusion network. In *International Conference on Spoken Language Processing*.

M. K. Tanenhous, M. Spivey-Knowlton, E. Eberhard, and J. Sedivy. 1995. Integration of visual and linguistic information during spoken language comprehension. In *Science*, volume 268, pages 1632–1634.

D. H. Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.

Multi-document summarization using A* search and discriminative training

Ahmet Aker **Trevor Cohn** **Robert Gaizauskas**

Department of Computer Science
University of Sheffield, Sheffield, S1 4DP, UK
{a.aker, t.cohn, r.gaizauskas}@dcs.shef.ac.uk

Abstract

In this paper we address two key challenges for extractive multi-document summarization: the search problem of finding the best scoring summary and the training problem of learning the best model parameters. We propose an A* search algorithm to find the best extractive summary up to a given length, which is both optimal and efficient to run. Further, we propose a discriminative training algorithm which directly maximises the quality of the best summary, rather than assuming a sentence-level decomposition as in earlier work. Our approach leads to significantly better results than earlier techniques across a number of evaluation metrics.

1 Introduction

Multi-document summarization aims to present multiple documents in form of a short summary. This short summary can be used as a replacement for the original documents to reduce, for instance, the time a reader would spend if she were to read the original documents. Following dominant trends in summarization research (Mani, 2001), we focus solely on extractive summarization which simplifies the summarization task to the problem of identifying a subset of units from the document collection (here sentences) which are concatenated to form the summary.

Most multi-document summarization systems define a model which assigns a score to a candidate summary based on the features of the sentences included in the summary. The research challenges are then twofold: 1) the *search* problem of finding the best scoring summary for a given document set, and 2) the *training* problem of learning the model parameters to best describe a training set consisting of pairs of document sets with model or reference summaries – typically human authored extractive or abstractive summaries.

Search is typically performed by a greedy algorithm which selects each sentence in decreasing order of model score until the desired summary length is reached (see, e.g., Saggion (2005)) or using heuristic strategies based on position in document or lexical clues (Edmundson, 1969; Brandow et al., 1995; Hearst, 1997; Ouyang et al., 2010).[1] We show in this paper that the search problem can be solved optimally and efficiently using A* search (Russell et al., 1995). Assuming the model only uses features local to each sentence in the summary, our algorithm finds the best scoring extractive summary up to a given length in words.

Framing summarization as search suggests that many of the popular training techniques are maximising the wrong objective. These approaches train a classifier, regression or ranking model to distinguish between good and bad sentences under an evaluation metric, e.g., ROUGE (Lin, 2004). The model is then used during search to find a summary composed of high scoring ('good') *sentences* (see for a review Ouyang et al. (2010)). However, there is a disconnect between the model used for training and the model used for prediction. In this paper we present a solution to this disconnect in the form of a training algorithm that optimises the full prediction model directly with the search algorithm intact. The training algorithm learns parameters such that

[1] Genetic algorithms have also been devised for solving the search problem (see, e.g., Riedhammer et al. (2008)), however these approaches do not guarantee optimality, nor are they efficient enough to be practicable for large datasets.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 482–491,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

the best scoring *whole summary* under the model has a high score under the evaluation metric. We demonstrate that this leads to significantly better test performance than a competitive baseline, to the tune of 3% absolute increase for ROUGE-1, -2 and -SU4.

The paper is structured as follows. Section 2 presents the summarization model. Next in section 3 we present an A* search algorithm for finding the best scoring (argmax) summary under the model with a constraint on the maximum summary length. We show that this algorithm performs search efficiently, even for very large document sets composed of many sentences. The second contribution of the paper is a new training method which directly optimises the summarization system, and is presented in section 4. This uses the minimum error-rate training (MERT) technique from machine translation (Och, 2003) to optimise the summariser's output to an arbitrary evaluation metric. Section 5 describes our experimental setup and section 6 the results. Finally we conclude in section 7.

2 Summarization Model

Extractive multi-document summarization aims to find the most important sentences from a set of documents, which are then collated and presented to the user in form of a short summary. Following the predominant approach to data-driven summarisation, we define a linear model which scores summaries as the weighted sum of their features,

$$s(\mathbf{y}|\mathbf{x}) = \Phi(\mathbf{x}, \mathbf{y}) \cdot \lambda \quad , \tag{1}$$

where $\mathbf{x}$ is the document set, composed of k sentences, $\mathbf{y} \subseteq \{1 \dots k\}$ are the set of selected sentence indices, $\Phi(\cdot, \cdot)$ is a feature function which returns a vector of features for the candidate summary and λ are the model parameters. We further assume that the features decompose with the sentences in the summary, $\Phi(\mathbf{x}, \mathbf{y}) = \sum_{i \in \mathbf{y}} \phi(x_i)$, and therefore the scoring function also decomposes along the same lines,

$$s(\mathbf{y}|\mathbf{x}) = \sum_{i \in \mathbf{y}} \phi(x_i) \cdot \lambda \quad . \tag{2}$$

While this assumption greatly simplifies inference, it does constrain the representative power of the model by disallowing global features, e.g., those which measure duplication in the summary.[2] Under this model, the search problem is to solve

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} s(\mathbf{y}|\mathbf{x}) \quad , \tag{3}$$

for which we develop a best-first algorithm using A* search, as described in section 3. The training challenge is to find the parameters, λ, to best model the training set. This is achieved by finding λ such that $\hat{\mathbf{y}}$ is similar to the gold standard summary according to an automatic evaluation metric, as described in section 4.

3 A* Search

The prediction problem is to find the best scoring extractive summary (see Equation 3) up to a given length, L. At first glance, this appears to be a simple problem that might be solved efficiently with a greedy algorithm, say by taking the sentences in order of decreasing score and stopping just before the summary exceeds the length threshold. However, the greedy algorithm cannot be guaranteed to find the best summary; to do so requires arbitrary backtracking to revise previous incorrect decisions.

The problem of constructing the summary can be considered a search problem in which we start with an empty summary and incrementally enlarge the summary by concatenating a sentence from our document set. The search graph starts with an empty summary (the starting state) and each outgoing edge adds a sentence to produce a subsequent state, and is assigned a score under the model. A goal state is any state with no more words than the given threshold. The summarisation problem is then equivalent to finding the best scoring path (summed over the edge scores) between the start state and a goal state.

The novel insight in our work is to use A* search (Russell et al., 1995) to solve the prediction problem. A* is a best-first search algorithm which can efficiently find the best scoring path or the n-best paths (unlike the greedy algorithm which is not optimal, and the backtracking variant which is not efficient). The search procedure requires a scoring function for each state, here $s(\mathbf{y}|\mathbf{x})$ from (2), and

[2] Our approach could be adapted to support global features, which would require changes to the heuristic for A* search to bound the score obtainable from the global features. This may incur an additional computational cost over a purely local feature model and perhaps also necessitate using beam search.

a heuristic function which estimates the additional score to get from a given state to a goal state. For the search to be optimal – guaranteed to find the best scoring path as the first solution – the heuristic must be *admissible*, meaning that it bounds from above the score for reaching a goal state. We present three different admissible heuristics later in this section, which bound the score with differing tightness and consequently different search cost.

Algorithm 1 presents A* search for our extractive summarisation model. Given a set of sentences to summary, a scoring and a heuristic function, it finds the best scoring summary. This is achieved by building the search graph incrementally, and storing each frontier state in a priority queue (line 1) which is sorted by the sum of the state's score and its heuristic. These states are popped off the queue (line 3) and expanded by adding a sentence, which is then added to the schedule (lines 8–14). We designate special finishing states using a boolean variable (the last entry in the tuple in lines 1, 7 and 12). Finishing states (with value *T*) denote ceasing to expand the summary, and consequently their scores do not include the heuristic component. Whenever one of these states is popped in line 2, we know that it outscores all competing hypotheses and therefore represents the optimal summary (because the heuristic is guaranteed to never underestimate the cost to a goal state from an unfinished state).[3] Note that in algorithm 1 we create the summary by building a list of sentence indices in sorted order to avoid spurious ambiguity which would unnecessarily expand the search space. The function length$(\mathbf{y}, \mathbf{x}) = \sum_{n \in \mathbf{y}}$ length(x_n) returns the length of sentences specified.

We now return to the problem of defining the heuristic function, $h(\mathbf{y}; \mathbf{x}, l)$ which provides an upper bound on the additional score achievable in reaching a goal state from state $\mathbf{y}$. We present three different variants of increasing fidelity, that is, that bound the cost to a goal state more tightly. Algorithm 2 is the simplest, which simply finds the maximum score per word from the set of unused sentences and then extrapolates this out over the remaining words available to the length threshold. In the algorithm, we use the shorthand $s_n = \phi(x_n) \cdot \lambda$ for sentence n's score, $l_n =$ length(x_n) for its length and $l_{\mathbf{y}} = \sum_{n \in \mathbf{y}} l_n$ for the total length of the current state (unfinished summary).

Algorithm 1 A* search for extractive summarization.

```
Require: set of sentences, $\mathbf{x} = x_1, \ldots, x_k$
Require: scoring function $s(\cdot)$
Require: heuristic function $h(\cdot)$
Require: summary length limit $L$
 1: schedule = [$(0, \emptyset, \mathrm{F})$]          {priority queue of triples}
                        {(A* score, sentence indices, done flag)}
 2: while schedule $\neq$ [] do
 3:     $v, \mathbf{y}, f \leftarrow$ pop(schedule)
 4:     if f = T then
 5:         return $\mathbf{y}$                              {success}
 6:     else
 7:         push(schedule, $(s(\mathbf{y}|\mathbf{x}), \mathbf{y}, \mathrm{T})$)
 8:         for $y \leftarrow [\max(\mathbf{y}) + 1, k]$ do
 9:             $\mathbf{y}' \leftarrow \mathbf{y} \cup y$
10:             if length$(\mathbf{y}', \mathbf{x}) \leq L$ then
11:                 $v' \leftarrow s(\mathbf{y}'|\mathbf{x}) + h(\mathbf{y}'; \mathbf{x}, l)$
12:                 push(schedule, $(v', \mathbf{y}', \mathrm{F})$)
13:             end if
14:         end for
15:     end if
16: end while
```

Algorithm 2 Uniform heuristic, $h_1(\mathbf{y}; \mathbf{x}, L)$

```
Require: $\mathbf{x}$ sorted in order of score/length
$n \leftarrow \max(\mathbf{y}) + 1$
return $(L - l_{\mathbf{y}}) \max\left(\frac{s_n}{l_n}, 0\right)$
```

The h_1 heuristic is overly simple in that it assumes we can 'reuse' a high scoring short sentence many times despite this being disallowed by the model. For this reason we develop an improved bound, h_2, in Algorithm 3. This incrementally adds each sentence in order of its score-per-word until the length limit is reached. If the limit is to be exceeded, the heuristic scales down the final sentence's score based on the fraction of words than can be used to reach the limit.

The fractional usage of the final sentence in h_2 could be considered overly optimistic, especially when the state has length just shy of the limit L. If the next best ranked sentence is a long one, then it will be used in the heuristic to over-estimate of the state. This is complicated to correct, and doing so exactly would require full backtracking which is intractable and would obviate the entire point of using A* search. Instead we use a subtle modification in h_3 (Alg. 4) which is equivalent to h_2 except in the

[3] To improve the efficiency of Algorithm 1 we make a small modification to avoid expanding every possible edge in step 8, of which there are $O(k)$ options. Instead we expand a small number (here, 3) at a time and defer the remaining items until later by inserting a special node into the schedule. These special nodes are represented using a third 'to-be-continued' state into the done flag.

Algorithm 3 Aggregated heuristic, $h_2(\mathbf{y}; \mathbf{x}, L)$

```
Require: $\mathbf{x}$ sorted in order of score/length
$v \leftarrow 0$
$l' \leftarrow l_{\mathbf{y}}$
for $n \in [\max(\mathbf{y}) + 1, k]$ do
    if $s_n \leq 0$ then
        return v
    end if
    if $l' + l_n \leq L$ then
        $l' \leftarrow l' + l_n$
        $v \leftarrow v + s_n$
    else
        return $v + \frac{l_n}{L-l'} s_n$
    end if
end for
return $v$
```

instance where the next best score/word sentence is too long, where it skips over these sentences until it finds the best scoring sentence that does fit. This helps to address the overestimate of h_2 and should therefore lead to a smaller search graph and faster runtime due to its early elimination of dead-ends.

Algorithm 4 Agg.+final heuristic, $h_3(\mathbf{y}; \mathbf{x}, L)$

```
Require: $\mathbf{x}$ sorted in order of score/length
$n \leftarrow \max(\mathbf{y}) + 1$
if $n \leq k \wedge s_n > 0$ then
    if $l_{\mathbf{y}} + l_n \leq L$ then
        return $h_2(\mathbf{y}; \mathbf{x}, L)$
    else
        for $m \in [n + 1, k]$ do
            if $l_{\mathbf{y}} + l_m \leq L$ then
                return $s_m \frac{L-l_{\mathbf{y}}}{l_m}$
            end if
        end for
    end if
end if
return 0
```

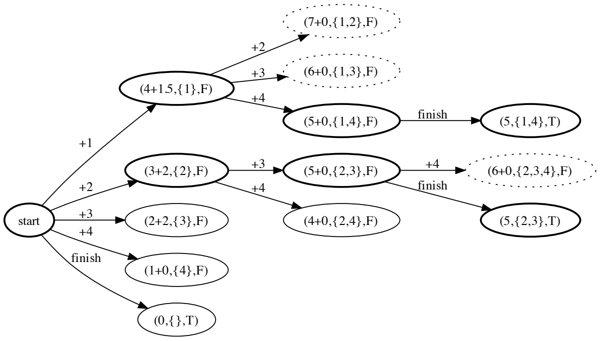


Figure 1: Example of the A* search graph created to find the two top scoring summaries of length ≤ 7 when summarising four sentences with scores of 4, 3, 2 and 1 respectively and lengths of 5, 4, 3 and 1 respectively. The h_1 heuristic was used and the score and heuristic scores are shown separately for clarity. Bold nodes were visited while dashed nodes were visited but found to exceed the length constraint.

The search process is illustrated in figure 1. When a node is visited in the search, if it satisfied the length constraint then the all its child nodes are added to the schedule. These nodes are scored with the score for the summary thus far plus a heuristic term. For example, the value of 4+1.5=5.5 for the $\{1\}$ node arises from a score of 4 plus a heuristic of $(7 - 5) \cdot \frac{3}{4} = 1.5$, reflecting the additional score that would arise if it were to use half of the next sentence to finish the summary. Note that in finding the best two summaries the search process did not need to instantiate the full search graph.

To test the efficacy of A* search with each of the different heuristic functions, we now present empirical runtime results. We used the training data as described in Section 5.2 and for each document set generated the 100-best summaries with word limit $L = 200$. Figure 2 shows the number of nodes and edges visited by A* search, reflecting the space and time cost of the algorithm, as a function of the number of sentences in the document set being summarised. All three heuristics shown an empirical increase in complexity that is roughly linear in the document size, although there are some notable outliers, particularly for the uniform heuristic. Surprisingly the aggregated heuristic, h_2, is not considerably more efficient than the uniform heuristic h_1, despite bounding the cost more precisely. However, the aggregated+final heuristic, h_3, consistently outperforms the other two methods. For this reason we have used h_3 in all subsequent experimentation.

4 Training

We frame the training problem as one of finding model parameters, λ, such that the predicted output, $\hat{\mathbf{y}}$ closely matches the gold standard, $\mathbf{r}$.[4] The quality of the match is measured using an automatic evaluation metric. We adopt the standard machine learning terminology of loss functions, which measure the degree of error in the prediction, $\Delta(\hat{\mathbf{y}}, \mathbf{r})$. In our case the accuracy is measured by the ROUGE

[4] The gold standard is typically an abstractive summary, and as such it is usually impossible for an extractive summarizer to match it exactly.

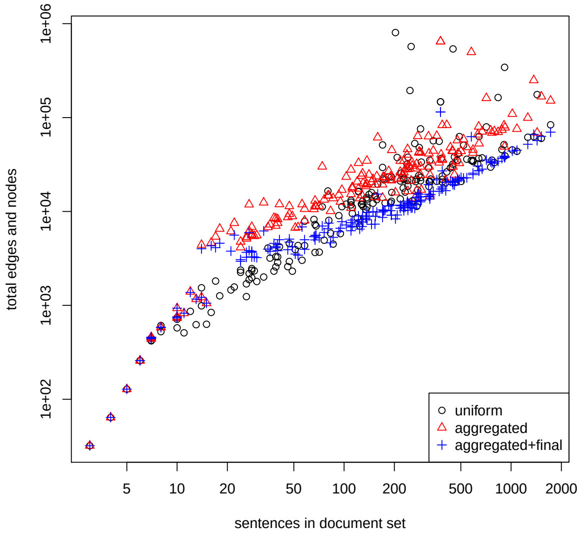


Figure 2: Efficiency of A* search search is roughly linear in the number of sentences in the document set. The y axis measures the search graph size in terms of the number of edges in the schedule and the number of nodes visited. Measured with the final parameters after training to optimise ROUGE-2 with the three different heuristics and expanding five nodes in each step.

score, R, and the loss is simply 1 - R. The training problem is to solve

$$\hat{\lambda} = \arg\min_{\lambda} \Delta(\hat{\mathbf{y}}, \mathbf{r}) \quad , \tag{4}$$

where with a slight abuse of notation, $\hat{\mathbf{y}}$ and $\mathbf{r}$ are taken to range over the corpus of many document-sets and summaries.

To optimise the weights we use the minimum error rate training (MERT) technique (Och, 2003), as used for training statistical machine translation systems. This approach is a first order optimization method using Powell search to find the parameters which minimise the loss on the training data. MERT requires n-best lists which it uses to approximate the full space of possible outcomes. We use the A* search algorithm to construct these n-best lists,[5] and use MERT to optimise the ROUGE score on the training set for the R-1, R-2 and R-SU4 variants of the metric.

5 Experimental settings

In this section we describe the features for which we learn weights. We also describe the input data used in training and testing.

5.1 Summarization system

The summarizer we use is an extractive, query-based multi-document summarization system. It is given two inputs: a query (place name) associated with an image and a set of documents. The summarizer uses the following features, as reported in previous work (Edmundson, 1969; Brandow et al., 1995; Radev et al., 2001; Conroy et al., 2005; Aker and Gaizauskas, 2009; Aker and Gaizauskas, 2010a):

- ***querySimilarity***: Sentence similarity to the query (cosine similarity over the vector representation of the sentence and the query).
- ***centroidSimilarity***: Sentence similarity to the centroid. The centroid is composed of the 100 most frequently occurring non stop words in the document collection (cosine similarity over the vector representation of the sentence and the centroid). For each word/term in the vector we store a value which is the product of the *term frequency* in the document and the *inverse document frequency*, a measurement of the term's distribution over the set of documents (Salton and Buckley, 1988).
- ***sentencePosition***: Position of the sentence within its document. The first sentence in the document gets the score 1 and the last one gets $\frac{1}{n}$ where n is the number of sentences in the document.
- ***inFirst5***: Binary feature indicating whether the sentence occurs is one of the first 5 sentences of the document.
- ***isStarter***: A sentence gets a binary score if it starts with the query term (e.g. *Westminster Abbey*, *The Westminster Abbey*, *The Westminster* or *The Abbey*) or with the object type, e.g. *The church*. We also allow gaps (up to four words) between *the* and the query/object type to capture cases such as *The most magnificent abbey*, etc.
- ***LMProb***: The probability of the sentence under a unigram language model. We trained a separate language model on Wikipedia articles about locations for each object type, e.g.,

[5] We used $n = 100$ in our experiments.

church, *bridge*, etc. When we generate a summary about a location of type church, for instance, then we apply the church language model on the related input documents related to the location.[6]
- ***sentenceCount***: Each sentence gets assigned a value of *1*. This feature is used to learn whether summaries with many sentences are better than summaries with few sentences or vice versa.
- ***wordCount***: Number of words in the summary, to decide whether the model should favor long summaries or short ones.

5.2 Data

For training and testing we use the freely available image description corpus described in Aker and Gaizauskas (2010b). The corpus is based around 289 images of static located objects (e.g *Eiffel Tower*, *Mont Blanc*) each with a manually assigned place name and object type category (e.g. *church*, *mountain*). For each place name there are up to four model summaries that were created manually after reading existing image descriptions taken from the *VirtualTourist* travel community web-site. Each summary contains a minimum of 190 and a maximum of 210 words. We divide this set of 289 place names into training and testing sets. Both sets are described in the following subsections.

Training We use 184 place names from the 289 set for training feature weights. For each training place name we gather all descriptions associated with it from *VirtualTourist*. We compute for each sentence in each description a ROUGE score by comparing the sentence to those included in the model summaries for that particular place name and retaining the highest score. Table 1 gives some details about this training data.

	Max	Min	Avg
Sentences/place	1724	3	260
Words/sentence	37	3	17

Table 1: The training input data contains 184 place names with 42333 sentences in total. The numbers in the columns give detail about the number of sentences for each place and the lengths of the sentences.

	Max	Min	Avg
Documents/place	20	5	12
Sentences/place	1716	15	132
Sentences/document	275	1	10
Words/sentence	211	1	20

Table 2: In domain test data. The numbers in the columns give detail about the number of documents (descriptions) for each place, number of sentences for each place and document (description) and the lengths of the sentences.

We use ROUGE as a metric to maximize because it is also used in DUC[7] and TAC.[8] However, it should be noted that any automatic metric could be used instead of ROUGE. In particular we use ROUGE 1 (R-1), ROUGE 2 (R-2) and ROUGE SU4 (R-SU4). R-1 and R-2 compute the number of uni-gram and bi-gram overlaps, respectively, between the automatic and model summaries. R-SU4 allows bi-grams to be composed of non-contiguous words, with a maximum of four words between the bi-grams.

Testing For testing purposes we use the rest of the place names (105) from the 289 place name set. For each place name we use a set of input documents, generate a summary from these documents using our summarizer and compare the results against model summaries of that place name using ROUGE. We experimented with two different input document types: out of domain and in domain.

The in domain documents are the VirtualTourist original image descriptions from which the model summaries were derived. As with the training set we take all place name descriptions for a particular place and use them as input documents to our summarizer. Table 2 summarizes these input documents.

The out of domain documents are retrieved from the web. Compared to the in domain documents these documents should more challenging to summarize because they will contain different kinds of documents to those seen in training. For each place name we retrieved the top ten related web-documents using the Yahoo! search engine with the place name as a query. The text from these documents is extracted using an HTML parser and passed to the summarizer. Table 3 gives an overview of this data.

[6] For our training and testing sets we manually assigned each location to its corresponding object type (Aker and Gaizauskas, 2009).

[7] http://duc.nist.gov/

[8] http://www.nist.gov/tac/

	Max	Min	Avg
Sentences/place	1773	55	328
Sentence/document	874	1	32
Words/sentence	236	1	21

Table 3: Out of domain test data. The numbers in the columns give detail about the number of sentences for each place and document and the lengths of the sentences.

Type	metric	R-1	R-2	R-SU4
wordLimit	R-1	**0.5792**	0.3176	0.3580
	R-2	0.5656	**0.3208**	0.3510
	R-SU4	0.5688	0.3197	**0.3585**
sentenceLimit	R-1	**0.5915**	0.3507	0.3881
	R-2	0.5783	**0.3601**	0.3890
	R-SU4	0.5870	0.3546	**0.3929**
regression	R-1	0.4993	0.1946	0.2448
	R-2	0.4833	0.1949	0.2413
	R-SU4	**0.5009**	**0.2031**	**0.2562**

Table 4: ROUGE scores obtained on the training data.

6 Results

To evaluate our approach we used two different assessment methods: ROUGE (Lin, 2004) and manual readability. In the following we present the results of each assessment.

6.1 Automatic Evaluation using ROUGE

We report results for training and testing. In both training and testing we distinguish between three different summaries: *wordLimit*, *sentenceLimit* and *regression*. *WordLimit* and *sentenceLimit* summaries are the ones generated using the model trained by MERT. As described in section 4 we trained the summariser using the A* search decoder to maximise the ROUGE score of the best scoring summaries. We used the heuristic function *h3* in A* search because it is the best performing heuristic, and 100-best lists. To experiment with different summary length conditions we differentiate between summaries with a word limit (*wordLimit*, set to 200 words) and summaries containing *N* number of sentences (*sentenceLimit*) as stop condition in A* search. We set *N* so that in both *wordLimit* and *sentenceLimit* summaries we obtain more or less the same number of words (because our training data contains on average 17 words for each word we set *N* to 12, 12*17=194). However, this is only the case in the training. In the testing for both *wordLimit* and *sentenceLimit* we generate summaries with the same word limit constraint which allows us to have a fair comparison between the ROUGE recall scores.

The *regression* summaries are our baseline. In these summaries the sentences are ranked based on the weighted features produced by Support Vector Regression (SVR).[9] Ouyang et al. (2010) use multi-document summarization and linear regression methods to rank sentences in the documents. As regression model they used SVR and showed that it out-performed classification and Learning To Rank methods on the DUC 2005 to 2007 data. For comparison purpose we use SVR as a baseline system for learning feature weights. It should be noted that these weights are learned based on single sentences. However, to have a fair comparison between all our summary types we use these weights to generate summaries using the A* search with the word limit as constraint. We do this for reporting both for training and testing results.

The results for training are shown in Table 4. The table shows ROUGE recall numbers obtained by comparing model summaries against automatically generated summaries on the training data. Because in training we used three different metrics (R-1, R-2, R-SU4) to train weights we report results for each of these three different ROUGE metrics.

In Table 4 we can see that the scores for *wordLimit* and *sentenceLimit* type summaries are always at maximum on the metric they were trained on (this can be observed by following the main diagonal of the result matrix). This confirms that MERT is maximizing the metric for which it was trained. However, this is not the case for regression results. The scores obtained with R-SU4 metric trained weights achieve higher scores on R-1 and R-2 compared to the scores obtained using weights trained on those metrics. This is most likely due to SVR being trained on sentences rather than over entire summaries, and thereby not adequately optimising the metric used for evaluation.

The results for testing are shown in Tables 5 and 6. As with the training setting we report ROUGE recall scores. We use the testing data described in section 5.2 for this setting. However, because we have two different input document sets we report separate results for each of these (Table 5 shows result for in domain data and Table 6 shows result for out

[9] We use the term *regression* to refer to SVR.

Type	metric	R-1	R-2	R-SU4
wordLimit	R-1	**0.3733**	**0.0842**	0.1399
	R-2	0.3731	**0.0842**	**0.1402**
	R-SU4	0.3627	0.0794	0.1340
sentenceLimit	R-1	0.3664	0.0774	0.1321
	R-2	0.3559	0.0717	0.1251
	R-SU4	0.3629	0.0778	0.1312
regression	R-1	0.3431	0.0669	0.1229
	R-2	0.2934	0.0560	0.1043
	R-SU4	0.3417	0.0668	0.1226

Table 5: ROUGE scores obtained on the testing data. The automated summaries are generated using the in domain input documents.

Type	metric	R-1	R-2	R-SU4
wordLimit	R-1	**0.3758**	0.0882	0.1421
	R-2	0.3755	**0.0895**	**0.1423**
	R-SU4	0.369	0.0812	0.137
sentenceLimit	R-1	0.3541	0.0693	0.1226
	R-2	0.3426	0.0638	0.1157
	R-SU4	0.3573	0.073	0.1251
regression	R-1	0.3392	0.0611	0.1179
	R-2	0.3422	0.0606	0.1164
	R-SU4	0.3413	0.0606	0.1176

Table 6: ROUGE scores obtained on the testing data. The automated summaries are generated using the out of domain input documents.

of domain data). Again as with the training setting we report results for the different metrics (R-1, R-2, R-SU4) separately.

From Table 5 we can see that the *wordLimit* summaries score highest compared to the other two types of summaries. This is different from the training results where *sentenceLimit* summary type summaries are the top scoring ones. As mentioned earlier the *sentenceLimit* summaries contain exactly 12 sentences, where on average each sentence in the training data has 17 words. We picked 12 sentences to achieve roughly the same word limit constraint ($12 \times 17 = 204$) so they can be compared to the *wordLimit* and *regression* type summaries. However, these *sentenceLimit* summaries have an average of 221 words, which explains the higher ROUGE recall scores seen in training compared to testing (where a 200 word limit was imposed).

The *wordLimit* summaries are significantly better than the scores from the other summary types irrespective of the evaluation metric.[10] It should be noted that these summaries are the only ones where the training and testing had the same condition in A* search concerning the summary word limit constraint. The scores in *sentenceLimit* type summaries are significantly lower than *wordLimit* summaries, despite using MERT to learn the weights. This shows that training the true model is critical for getting good accuracy. The *regression* type summaries achieved the worst ROUGE metric scores. The weights used to generate these summaries were trained on single sentences using SVR. These results indicate that if the goal is to generate high scoring summaries under a length limit in testing, then the same constraint should also be used in training.

From Table 5 and 6 we can see that the summaries obtained from VirtualTourist captions (in domain data) score roughly the same as the summaries generated using web-documents (out of domain data) as input. A possible explanation is that in many cases the VirtualTourist original captions contain text from Wikipedia articles, which are also returned as results from the web search. Therefore the web-document sets included similar content to the VirtualTourist captions.

6.2 Manual Evaluation

We also evaluated our summaries using a readability assessment as in DUC and TAC. DUC and TAC manually assess the quality of automatically generated summaries by asking human subjects to score each summary using five criteria – *grammaticality, redundancy, clarity, focus* and *coherence* criteria. Each criterion is scored on a five point scale with high scores indicating a better result (Dang, 2005).

For this evaluation we used the best scoring summaries from the *wordLimit* summary type (R-1, R-2 and R-SU4) generated using web-documents (out of domain documents) as input. We also evaluate the *regression* summary types generated using the same input documents to investigate the correlation between high and low ROUGE metric scores to manual evaluation ones. From the *regression* summary type we only use summaries under the *R2* and *RSU4* trained models.

In total we evaluated five different summary types (three from *wordLimit* and two from *regression*). For each type we randomly selected 30 place names and asked three people to assess the summaries for these place names. Each person was shown all 150

[10] Significance is reported at level $p < 0.001$. We used Wilcoxson signed ranked test to perform significance.

Criterion	wordLimit			regression	
	R1	R2	RSU4	R2	RSU4
clarity	4.03	3.92	3.99	3.00	2.92
coherence	3.31	3.06	2.99	2.12	1.88
focus	3.79	3.56	3.54	2.44	2.29
grammaticality	4.21	4.13	4.13	3.93	3.87
redundancy	4.19	4.33	4.41	4.47	4.44

Table 7: Manual evaluation results for the *wordLimit* (R1, R2, RSU4) and *regression* (R2, RSU4) summary types. The numbers in the columns are the average scores.

summaries (30 from each summary type) in a random way and was asked to assess them according to the DUC and TAC manual assessment scheme. The results are shown in Table 7.[11]

From Table 7 we can see that overall the *wordLimit* type summaries perform better than the *regression* ones. For each metric in *regression* summary types (R-2 and R-SU4) we compute the significance of the difference with the same metrics in *wordLimit* summary types.[12] The results for the *clarity*, *coherence* and *focus* criteria in *wordLimit* summaries are significantly better than in *regression* ones (p<0.001) irrespective of the training metric. These results concur with the automatic evaluation results as described in section 6.1. However, this is not the case for the *grammaticality* and *redundancy* criteria. Although in *regression* type summaries the scores for the *grammaticality* criterion are lower than those in *wordLimit* summaries the difference is not significant. Furthermore, we can see that the *redundancy* scores for *regression* summaries are slightly higher than those for *wordLimit* summaries.

One reason for these differences might be the way we trained feature weights for *wordLimit* and *regression* summaries. As mentioned above, feature weights for *wordLimit* summaries are trained using summaries with a specific word limit constraint, whereas the weights for the *regression* summaries are learned using single sentences. Maximizing the ROUGE metrics using "final or output like summaries" will lead to a higher content agreement between the training and the model summaries whereas this is not guaranteed with single sentences. With single sentences we have only a guarantee for high content overlap between single training and model sentences. However, when these sentences are combined into summaries it is not guaranteed that these summaries will also have high content overlap with the entire model ones. Therefore we believe if there is a high content agreement between the training and model summaries this could lead to more readable summaries. However, as we can see from Table 7 this hypothesis does not hold for all criteria. In case of the *redundancy* criterion we have compared to *wordLimit* summary type high scores in *regression* summaries although *wordLimit* summaries are significantly better than *regression* ones when it concerns the ROUGE scores. Thus it is likely that by aggressively optimising the ROUGE metric the model learns to game the metric, which does not penalise redundancy in the summaries. As such it may no longer possible to extrapolate trends from earlier correlation studies against human judgements (Lin, 2004).

To minimize redundancy in summaries it is necessary to also take into consideration global features addressing the linguistic aspects of the summaries. Furthermore, instead of ROUGE recall scores which do not take the repetition of information into consideration, ROUGE precision scores could be used as a metric in order to minimize the redundant content in the summaries.

7 Conclusion

In this paper we have proposed an A* search approach for generating a summary from a ranked list of sentences and learning feature weights for a feature based extractive multi-document summarization system. We developed an algorithm to learn optimize an arbitrary metric and showed that our approach significantly outperforms state of the art techniques. Furthermore, we highlighted the importance of uniformity in training and testing and argued that if the goal is to generate high scoring summaries under a length limit in testing, then the same constraint should also be used in training.

In this paper we experimented with sentence-local features. In the future we plan to expand this feature set with global features, especially ones mea-

[11] We computed the agreement between the users using intra class correlation with Cronbach's Alpha where the correlation coefficient ranges between 0 and 1. Numbers close to 1 indicate high correlation and numbers close to 0 indicate low correlation. For the clarity criterion the assessors' correlation coefficient is 0.547, for coherence 0.687, for focus 0.688, for grammaticality 0.232 and for redundancy 0.453.

[12] We compute significance test for the manual evaluation results using χ square.

suring lexical diversity in the summaries to reduce the redundancy in them. We will investigate various ways of incorporating these global features into our A* search. However this will incur an additional computational cost over a purely local feature model and therefore may necessitate using an approximate beam search. We also plan to investigate using other metrics in training in order to reduce redundant information in the summaries. Finally, we have made our summarizer publicly available as open-source software.[13]

References

A. Aker and R. Gaizauskas. 2009. Summary Generation for Toponym-Referenced Images using Object Type Language Models. *International Conference on Recent Advances in Natural Language Processing (RANLP) September 14-16, 2009, Borovets, Bulgaria*.

A. Aker and R. Gaizauskas. 2010a. Generating image descriptions using dependency relational patterns. *Proc. of the ACL 2010, Upsala, Sweden*.

A. Aker and R. Gaizauskas. 2010b. Model Summaries for Location-related Images. In *Proc. of the LREC-2010 Conference*.

R. Brandow, K. Mitze, and L.F. Rau. 1995. Automatic condensation of electronic publications by sentence selection* 1. *Information Processing & Management*, 31(5):675–685.

J.M. Conroy, J.D. Schlesinger, and J.G. Stewart. 2005. CLASSY query-based multi-document summarization. *Proc. of the 2005 Document Understanding Workshop, Boston*.

H.T. Dang. 2005. Overview of DUC 2005. *DUC 05 Workshop at HLT/EMNLP*.

H. Edmundson, P. 1969. New Methods in Automatic Extracting. *Journal of the Association for Computing Machinery*, 16:264–285.

M.A. Hearst. 1997. TextTiling: segmenting text into multi-paragraph subtopic passages. *Computational linguistics*, 23(1):33–64.

C-Y. Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out: Proc. of the ACL-04 Workshop*, pages 74–81.

I. Mani. 2001. *Automatic Summarization*. John Benjamins Publishing Company.

F.J. Och. 2003. Minimum error rate training in statistical machine translation. *Proc. of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, page 167.

Y. Ouyang, W. Li, S. Li, and Q. Lu. 2010. Applying regression models to query-focused multi-document summarization. *Information Processing & Management*.

D.R. Radev, S. Blair-Goldensohn, and Z. Zhang. 2001. Experiments in single and multi-document summarization using MEAD. *Document Understanding Conference*.

K. Riedhammer, D. Gillick, B. Favre, and D. Hakkani-T ”ur. 2008. Packing the meeting summarization knapsack. *Proc. Interspeech, Brisbane, Australia*.

S.J. Russell, P. Norvig, J.F. Canny, J. Malik, and D.D. Edwards. 1995. *Artificial intelligence: a modern approach*. Prentice hall Englewood Cliffs, NJ.

H. Saggion. 2005. Topic-based Summarization at DUC 2005. *Document Understanding Conference (DUC05)*.

G. Salton and C. Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing and Management: an International Journal*, 24(5):513–523.

[13] Available from `http://www.dcs.shef.ac.uk/~tcohn/a-star`

A Multi-Pass Sieve for Coreference Resolution

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers,
Mihai Surdeanu, Dan Jurafsky, Christopher Manning
Computer Science Department
Stanford University, Stanford, CA 94305
{kr,heeyoung,sudarshn,natec,mihais,jurafsky,manning}@stanford.edu

Abstract

Most coreference resolution models determine if two mentions are coreferent using a single function over a set of constraints or features. This approach can lead to incorrect decisions as lower precision features often overwhelm the smaller number of high precision ones. To overcome this problem, we propose a simple coreference architecture based on a sieve that applies tiers of deterministic coreference models one at a time from highest to lowest precision. Each tier builds on the previous tier's entity cluster output. Further, our model propagates global information by sharing attributes (e.g., gender and number) across mentions in the same cluster. This cautious sieve guarantees that stronger features are given precedence over weaker ones and that each decision is made using all of the information available at the time. The framework is highly modular: new coreference modules can be plugged in without any change to the other modules. In spite of its simplicity, our approach outperforms many state-of-the-art supervised and unsupervised models on several standard corpora. This suggests that sieve-based approaches could be applied to other NLP tasks.

1 Introduction

Recent work on coreference resolution has shown that a rich feature space that models lexical, syntactic, semantic, and discourse phenomena is crucial to successfully address the task (Bengston and Roth, 2008; Haghighi and Klein, 2009; Haghighi and Klein, 2010). When such a rich representation is available, even a simple deterministic model can achieve state-of-the-art performance (Haghighi and Klein, 2009).

By and large most approaches decide if two mentions are coreferent using a single function over all these features and information local to the two mentions.[1] This is problematic for two reasons: (1) lower precision features may overwhelm the smaller number of high precision ones, and (2) local information is often insufficient to make an informed decision. Consider this example:

The second attack occurred after some rocket firings aimed, apparently, toward [the israelis], apparently in retaliation. [we]'re checking our facts on that one. ... the president, quoted by ari fleischer, his spokesman, is saying he's concerned the strike will undermine efforts by palestinian authorities to bring an end to terrorist attacks and does not contribute to the security of [israel].

Most state-of-the-art models will incorrectly link *we* to *the israelis* because of their proximity and compatibility of attributes (both *we* and *the israelis* are plural). In contrast, a more cautious approach is to first cluster *the israelis* with *israel* because the demonymy relation is highly precise. This initial clustering step will assign the correct animacy attribute (`inanimate`) to the corresponding geo-political entity, which will prevent the incorrect merging with the mention *we* (`animate`) in later steps.

We propose an unsupervised sieve-like approach to coreference resolution that addresses these is-

[1] As we will discuss below, some approaches use an additional component to infer the overall best mention clusters for a document, but this is still based on confidence scores assigned using local information.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 492–501,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

sues. The approach applies tiers of coreference models one at a time from highest to lowest precision. Each tier builds on the entity clusters constructed by previous models in the sieve, guaranteeing that stronger features are given precedence over weaker ones. Furthermore, each model's decisions are richly informed by sharing attributes across the mentions clustered in earlier tiers. This ensures that each decision uses all of the information available at the time. We implemented all components in our approach using only deterministic models. All our components are unsupervised, in the sense that they do not require training on gold coreference links.

The contributions of this work are the following:

- We show that a simple scaffolding framework that deploys strong features through tiers of models performs significantly better than a single-pass model. Additionally, we propose several simple, yet powerful, new features.
- We demonstrate how far one can get with simple, deterministic coreference systems that do not require machine learning or detailed semantic information. Our approach outperforms most other unsupervised coreference models and several supervised ones on several datasets.
- Our modular framework can be easily extended with arbitrary models, including statistical or supervised models. We believe that our approach also serves as an ideal platform for the development of future coreference systems.

2 Related Work

This work builds upon the recent observation that strong features outweigh complex models for coreference resolution, in both supervised and unsupervised learning setups (Bengston and Roth, 2008; Haghighi and Klein, 2009). Our work reinforces this observation, and extends it by proposing a novel architecture that: (a) allows easy deployment of such features, and (b) infuses global information that can be readily exploited by these features or constraints.

Most coreference resolution approaches perform the task by aggregating local decisions about pairs of mentions (Bengston and Roth, 2008; Finkel and Manning, 2008; Haghighi and Klein, 2009; Stoyanov, 2010). Two recent works that diverge from this pattern are Culotta et al. (2007) and Poon and Domingos (2008). They perform coreference resolution jointly for all mentions in a document, using first-order probabilistic models in either supervised or unsupervised settings. Haghighi and Klein (2010) propose a generative approach that models entity clusters explicitly using a mostly-unsupervised generative model. As previously mentioned, our work is not constrained by first-order or Bayesian formalisms in how it uses cluster information. Additionally, the deterministic models in our tiered model are significantly simpler, yet perform generally better than the complex inference models proposed in these works.

From a high level perspective, this work falls under the theory of shaping, defined as a "method of successive approximations" for learning (Skinner, 1938). This theory is known by different names in many NLP applications: Brown et al. (1993) used simple models as "stepping stones" for more complex word alignment models; Collins (1999) used "cautious" decision list learning for named entity classification; Spitkovsky et al. (2010) used "baby steps" for unsupervised dependency parsing, etc. To the best of our knowledge, we are the first to apply this theory to coreference resolution.

3 Description of the Task

Intra-document coreference resolution clusters together textual mentions within a single document based on the underlying referent entity. Mentions are usually noun phrases (NPs) headed by nominal or pronominal terminals. To facilitate comparison with most of the recent previous work, we report results using gold mention boundaries. However, our approach does not make any assumptions about the underlying mentions, so it is trivial to adapt it to predicted mention boundaries (e.g., see Haghighi and Klein (2010) for a simple mention detection model).

3.1 Corpora

We used the following corpora for development and evaluation:

- **ACE2004-ROTH-DEV**[2] – development split of Bengston and Roth (2008), from the corpus used in the 2004 Automatic Content Extraction (ACE) evaluation. It contains 68 documents and 4,536 mentions.

[2] We use the same corpus names as (Haghighi and Klein, 2009) to facilitate comparison with previous work.

- **ACE2004-CULOTTA-TEST** – partition of ACE 2004 corpus reserved for testing by several previous works (Culotta et al., 2007; Bengston and Roth, 2008; Haghighi and Klein, 2009). It consists of 107 documents and 5,469 mentions.
- **ACE2004-NWIRE** – the newswire subset of the ACE 2004 corpus, utilized by Poon and Domingos (2008) and Haghighi and Klein (2009) for testing. It contains 128 documents and 11,413 mentions.
- **MUC6-TEST** – test corpus from the sixth Message Understanding Conference (MUC-6) evaluation. It contains 30 documents and 2,068 mentions.

We used the first corpus (ACE2004-ROTH-DEV) for development. The other corpora are reserved for testing. We parse all documents using the Stanford parser (Klein and Manning, 2003). The syntactic information is used to identify the mention head words and to define the ordering of mentions in a given sentence (detailed in the next section). For a fair comparison with previous work, we do not use gold named entity labels or mention types but, instead, take the labels provided by the Stanford named entity recognizer (NER) (Finkel et al., 2005).

3.2 Evaluation Metrics

We use three evaluation metrics widely used in the literature: (a) pairwise F1 (Ghosh, 2003) – computed over mention pairs in the same entity cluster; (b) MUC (Vilain et al., 1995) – which measures how many predicted clusters need to be merged to cover the gold clusters; and (c) B^3 (Amit and Baldwin, 1998) – which uses the intersection between predicted and gold clusters for a given mention to mark correct mentions and the sizes of the the predicted and gold clusters as denominators for precision and recall, respectively. We refer the interested reader to (X. Luo, 2005; Finkel and Manning, 2008) for an analysis of these metrics.

4 Description of the Multi-Pass Sieve

Our sieve framework is implemented as a succession of independent coreference models. We first describe how each model selects candidate mentions, and then describe the models themselves.

4.1 Mention Processing

Given a mention m_i, each model may either decline to propose a solution (in the hope that one of the subsequent models will solve it) or deterministically select a single best antecedent from a list of previous mentions $m_1, \ldots, m_{i-1}$. We sort candidate antecedents using syntactic information provided by the Stanford parser, as follows:

Same Sentence – Candidates in the same sentence are sorted using left-to-right breadth-first traversal of syntactic trees (Hobbs, 1977). Figure 1 shows an example of candidate ordering based on this traversal. The left-to-right ordering favors subjects, which tend to appear closer to the beginning of the sentence and are more probable antecedents. The breadth-first traversal promotes syntactic salience by ranking higher noun phrases that are closer to the top of the parse tree (Haghighi and Klein, 2009). If the sentence containing the anaphoric mention contains multiple clauses, we repeat the above heuristic separately in each `S*` constituent, starting with the one containing the mention.

Previous Sentence – For all nominal mentions we sort candidates in the previous sentences using right-to-left breadth-first traversal. This guarantees syntactic salience and also favors document proximity. For pronominal mentions, we sort candidates in previous sentences using left-to-right traversal in order to favor subjects. Subjects are more probable antecedents for pronouns (Kertz et al., 2006). For example, this ordering favors the correct candidate (*pepsi*) for the mention *they*:

> *[pepsi] says it expects to double [quaker]'s snack food growth rate. after a month-long courtship, [they] agreed to buy quaker oats...*

In a significant departure from previous work, each model in our framework gets (possibly incomplete) clustering information for each mention from the earlier coreference models in the multi-pass system. In other words, each mention m_i may already be assigned to a cluster C_j containing a set of mentions: $C_j = \{m_1^j, \ldots, m_k^j\}$; $m_i \in C_j$. Unassigned mentions are unique members of their own cluster. We use this information in several ways:

Attribute sharing – Pronominal coreference resolution (discussed later in this section) is severely af-

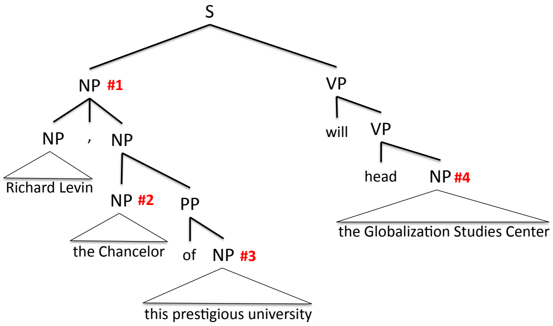


Figure 1: Example of left-to-right breadth-first tree traversal. The numbers indicate the order in which the NPs are visited.

fected by missing attributes (which introduce precision errors because incorrect antecedents are selected due to missing information) and incorrect attributes (which introduce recall errors because correct links are not generated due to attribute mismatch between mention and antecedent). To address this issue, we perform a union of all mention attributes (e.g., number, gender, animacy) in a given cluster and share the result with all cluster mentions. If attributes from different mentions contradict each other we maintain all variants. For example, our naive number detection assigns `singular` to the mention *a group of students* and `plural` to *five students*. When these mentions end up in the same cluster, the resulting number attributes becomes the set {`singular`, `plural`}. Thus this cluster can later be merged with both singular and plural pronouns.

Mention selection – Traditionally, a coreference model attempts to resolve every mention in the text, which increases the likelihood of errors. Instead, in each of our models, we exploit the cluster information received from the previous stages by resolving only mentions that are currently first in textual order in their cluster. For example, given the following ordered list of mentions, $\{m_1^1, m_2^2, m_3^2, m_4^3, m_5^1, m_6^2\}$, where the superscript indicates cluster id, our model will attempt to resolve only m_2^2 and m_4^3. These two are the only mentions that have potential antecedents and are currently marked as the first mentions in their clusters. The intuition behind this heuristic is two-fold. First, early cluster mentions are usually better defined than subsequent ones, which are likely to have fewer modifiers or are pronouns (Fox, 1993). Several of our models use this modifier information. Second, by definition, first mentions appear closer to the beginning of the document, hence there are fewer antecedent candidates to select from, and fewer opportunities to make a mistake.

Search Pruning – Finally, we prune the search space using *discourse salience*. We disable coreference for first cluster mentions that: (a) are or start with indefinite pronouns (e.g., *some*, *other*), or (b) start with indefinite articles (e.g., *a*, *an*). One exception to this rule is the model deployed in the first pass; it only links mentions if their entire extents match exactly. This model is triggered for all nominal mentions regardless of discourse salience, because it is possible that indefinite mentions are repeated in a document when concepts are discussed but not instantiated, e.g., *a sports bar* below:

> *Hanlon, a longtime Broncos fan, thinks it is the perfect place for [a sports bar] and has put up a blue-and-orange sign reading, "Wanted Broncos Sports Bar On This Site." . . . In a Nov. 28 letter, Proper states "while we have no objection to your advertising the property as a location for [a sports bar], using the Broncos' name and colors gives the false impression that the bar is or can be affiliated with the Broncos."*

4.2 The Modules of the Multi-Pass Sieve

We now describe the coreference models implemented in the sieve. For clarity, we summarize them in Table 1 and show the cumulative performance as they are added to the sieve in Table 2.

4.2.1 Pass 1 - Exact Match

This model links two mentions only if they contain exactly the same extent text, including modifiers and determiners, e.g., *the Shahab 3 ground-ground missile*. As expected, this model is extremely precise, with a pairwise precision over 96%.

4.2.2 Pass 2 - Precise Constructs

This model links two mentions if any of the conditions below are satisfied:

Appositive – the two nominal mentions are in an appositive construction, e.g., *[Israel's Deputy Defense Minister], [Ephraim Sneh] , said* . . . We use the same syntactic rules to detect appositions as Haghighi and Klein (2009).

Pass	Type	Features
1	N	exact extent match
2	N,P	appositive | predicate nominative | role appositive | relative pronoun | acronym | demonym
3	N	cluster head match & word inclusion & compatible modifiers only & not i-within-i
4	N	cluster head match & word inclusion & not i-within-i
5	N	cluster head match & compatible modifiers only & not i-within-i
6	N	relaxed cluster head match & word inclusion & not i-within-i
7	P	pronoun match

Table 1: Summary of passes implemented in the sieve. The Type column indicates the type of coreference in each pass: N – nominal or P – pronominal. & and | indicate conjunction and disjunction of features, respectively.

Predicate nominative – the two mentions (nominal or pronominal) are in a copulative subject-object relation, e.g., *[The New York-based College Board] is [a nonprofit organization that administers the SATs and promotes higher education]* (Poon and Domingos, 2008).

Role appositive – the candidate antecedent is headed by a noun and appears as a modifier in an NP whose head is the current mention, e.g., *[[actress] Rebecca Schaeffer]*. This feature is inspired by Haghighi and Klein (2009), who triggered it only if the mention is labeled as a person by the NER. We constrain this heuristic more in our work: we allow this feature to match only if: (a) the mention is labeled as a person, (b) the antecedent is animate (we detail animacy detection in Pass 7), and (c) the antecedent's gender is not neutral.

Relative pronoun – the mention is a relative pronoun that modifies the head of the antecedent NP, e.g., *[the finance street [which] has already formed in the Waitan district]*.

Acronym – both mentions are tagged as NNP and one of them is an acronym of the other, e.g., *[Agence France Presse] . . . [AFP]*. We use a simple acronym detection algorithm, which marks a mention as an acronym of another if its text equals the sequence of upper case characters in the other mention. We will adopt better solutions for acronym detection in future work (Schwartz, 2003).

Demonym – one of the mentions is a demonym of the other, e.g., *[Israel] . . . [Israeli]*. For demonym detection we use a static list of countries and their gentilic forms from Wikipedia.[3]

All the above features are extremely precise. As shown in Table 2 the pairwise precision of the sieve after adding these features is over 95% and recall increases 5 points.

4.2.3 Pass 3 - Strict Head Matching

Linking a mention to an antecedent based on the naive matching of their head words generates a lot of spurious links because it completely ignores possibly incompatible modifiers (Elsner and Charniak, 2010). For example, *Yale University* and *Harvard University* have similar head words, but they are obviously different entities. To address this issue, this pass implements several features that must all be matched in order to yield a link:

Cluster head match – the mention head word matches *any* head word in the antecedent cluster. Note that this feature is actually more relaxed than naive head matching between mention and antecedent candidate because it is satisfied when the mention's head matches the head of any entity in the candidate's cluster. We constrain this feature by enforcing a conjunction with the features below.

Word inclusion – all the non-stop[4] words in the mention cluster are included in the set of non-stop words in the cluster of the antecedent candidate. This heuristic exploits the property of discourse that it is uncommon to introduce novel information in later mentions (Fox, 1993). Typically, mentions of the same entity become shorter and less informative as the narrative progresses. For example, the two mentions in *. . . intervene in the [Florida Supreme Court]'s move . . . does look like very dramatic change made by [the Florida court]* point to the same entity, but the two mentions in the text below belong to different clusters:

The pilot had confirmed . . . he had turned onto

[3] http://en.wikipedia.org/wiki/List_of_adjectival_and_demonymic_forms_of_place_names

[4] Our stop word list includes person titles as well.

	MUC			B^3			Pairwise		
Passes	P	R	F1	P	R	F1	P	R	F1
{1}	95.9	31.8	47.8	99.1	53.4	69.4	96.9	15.4	26.6
{1,2}	95.4	43.7	59.9	98.5	58.4	73.3	95.7	20.6	33.8
{1,2,3}	92.1	51.3	65.9	96.7	62.9	76.3	91.5	26.8	41.5
{1,2,3,4}	91.7	51.9	66.3	96.5	63.5	76.6	91.4	27.8	42.7
{1,2,3,4,5}	91.1	52.6	66.7	96.1	63.9	76.7	90.3	28.4	43.2
{1,2,3,4,5,6}	89.5	53.6	67.1	95.3	64.5	76.9	88.8	29.2	43.9
{1,2,3,4,5,6,7}	83.7	74.1	78.6	88.1	74.2	80.5	80.1	51.0	62.3

Table 2: Cumulative performance on development (ACE2004-ROTH-DEV) as passes are added to the sieve.

[the correct runway] but pilots behind him say
he turned onto [the wrong runway].

Compatible modifiers only – the mention's modifiers are all included in the modifiers of the antecedent candidate. This feature models the same discourse property as the previous feature, but it focuses on the two individual mentions to be linked, rather than their entire clusters. For this feature we only use modifiers that are nouns or adjectives.

Not i-within-i – the two mentions are not in an i-within-i construct, i.e., one cannot be a child NP in the other's NP constituent (Haghighi and Klein, 2009).

This pass continues to maintain high precision (91% pairwise) while improving recall significantly (over 6 points pairwise and almost 8 points MUC).

4.2.4 Passes 4 and 5 - Variants of Strict Head

Passes 4 and 5 are different relaxations of the feature conjunction introduced in Pass 3, i.e., Pass 4 removes the `compatible modifiers only` feature, while Pass 5 removes the `word inclusion` constraint. All in all, these two passes yield an improvement of 1.7 pairwise F1 points, due to recall improvements. Table 2 shows that the `word inclusion` feature is more precise than `compatible modifiers only`, but the latter has better recall.

4.2.5 Pass 6 - Relaxed Head Matching

This pass relaxes the cluster head match heuristic by allowing the mention head to match any word in the cluster of the candidate antecedent. For example, this heuristic matches the mention *Sanders* to a cluster containing the mentions {*Sauls*, *the judge*, *Circuit Judge N. Sanders Sauls*}. To maintain high precision, this pass requires that both mention and antecedent be labeled as named entities and the types coincide. Furthermore, this pass implements a conjunction of the above features with `word inclusion` and `not i-within-i`. This pass yields less than 1 point improvement in most metrics.

4.2.6 Pass 7 - Pronouns

With one exception (Pass 2), all the previous coreference models focus on nominal coreference resolution. However, it would be incorrect to say that our framework ignores pronominal coreference in the first six passes. In fact, the previous models prepare the stage for pronominal coreference by constructing precise clusters with shared mention attributes. These are crucial factors for pronominal coreference.

Like previous work, we implement pronominal coreference resolution by enforcing agreement constraints between the coreferent mentions. We use the following attributes for these constraints:

Number – we assign number attributes based on: (a) a static list for pronouns; (b) NER labels: mentions marked as a named entity are considered singular with the exception of organizations, which can be both singular or plural; (c) part of speech tags: `NN*S` tags are plural and all other `NN*` tags are singular; and (d) a static dictionary from (Bergsma and Lin, 2006).

Gender – we assign gender attributes from static lexicons from (Bergsma and Lin, 2006; Ji and Lin, 2009).

Person – we assign person attributes only to pronouns. However, we do not enforce this constraint when linking two pronouns if one appears within quotes. This is a simple heuristic for speaker detection, e.g., *I* and *she* point to the same person in

"[I] voted my conscience," [she] said.

Animacy – we set animacy attributes using: (a) a static list for pronouns; (b) NER labels, e.g., PERSON is animate whereas LOCATION is not; and (c) a dictionary boostrapped from the web (Ji and Lin, 2009).

NER label – from the Stanford NER.

If we cannot detect a value, we set attributes to unknown and treat them as wildcards, i.e., they can match any other value.

This final model raises the pairwise recall of our system almost 22 percentage points, with only an 8 point drop in pairwise precision. Table 2 shows that similar behavior is measured for all other metrics. After all passes have run, we take the transitive closure of the generated clusters as the system output.

5 Experimental Results

We present the results of our approach and other relevant prior work in Table 3. We include in the table all recent systems that report results under the same conditions as our experimental setup (i.e., using gold mentions) and use the same corpora. We exclude from this analysis two notable works that report results only on a version of the task that includes finding mentions (Haghighi and Klein, 2010; Stoyanov, 2010). The Haghighi and Klein (2009) numbers have two variants: with semantics (+S) and without (−S). To measure the contribution of our multi-pass system, we also present results from a single-pass variant of our system that uses all applicable features from the multi-pass system (marked as "single pass" in the table).

Our sieve model outperforms all systems on two out of the four evaluation corpora (ACE2004-ROTH-DEV and ACE2004-NWIRE), on all metrics. On the corpora where our model is not best, it ranks a close second. For example, in ACE2004-CULOTTA-TEST our system has a B^3 F1 score only .4 points lower than Bengston and Roth (2008) and it outperforms all unsupervised approaches. In MUC6-TEST, our sieve's B^3 F1 score is 1.8 points lower than Haghighi and Klein (2009) +S, but it outperforms a supervised system that used gold named entity labels. Finally, the multi-pass architecture always beats the equivalent single-pass system with its contribution ranging between 1 and 4 F1 points depending on the corpus and evaluation metric.

Our approach has the highest precision on all corpora, regardless of evaluation metric. We believe this is particularly useful for large-scale NLP applications that use coreference resolution components, e.g., question answering or information extraction. These applications can generally function without coreference information so it is beneficial to provide such information only when it is highly precise.

6 Discussion

6.1 Comparison to Previous Work

The sieve model outperforms all other systems on at least two test sets, even though most of the other models are significantly richer. Amongst the comparisons, several are supervised (Bengston and Roth, 2008; Finkel and Manning, 2008; Culotta et al., 2007). The system of Haghighi and Klein (2009) +S uses a lexicon of semantically-compatible noun pairs acquired transductively, i.e., with knowledge of the mentions in the test set. Our system does not rely on labeled corpora for training (like supervised approaches) nor access to corpora during testing (like Haghighi and Klein (2009)).

The system that is closest to ours is Haghighi and Klein (2009) −S. Like us, they use a rich set of features and deterministic decisions. However, theirs is a single-pass model with a smaller feature set (no cluster-level, acronym, demonym, or animacy information). Table 3 shows that on the two corpora where results for this system are available, we outperform it considerably on all metrics. To understand if the difference is due to the multi-pass architecture or the richer feature set we compared (Haghighi and Klein, 2009) −S against both our multi-pass system and its single-pass variant. The comparison indicates that both these contributions help: our single-pass system outperforms Haghighi and Klein (2009) consistently, and the multi-pass architecture further improves the performance of our single-pass system between 1 and 4 F1 points, depending on the corpus and evaluation metric.

6.2 Semantic Head Matching

Recent unsupervised coreference work from Haghighi and Klein (2009) included a novel semantic component that matched related head words (e.g., *AOL* is a *company*) learned from select

	MUC			B^3			Pairwise		
	P	R	F1	P	R	F1	P	R	F1
ACE2004-ROTH-DEV									
This work (sieve)	**83.7**	**74.1**	**78.6**	**88.1**	74.2	**80.5**	**80.1**	51.0	**62.3**
This work (single pass)	82.2	72.6	77.1	86.8	72.6	79.1	76.0	47.6	58.5
Haghighi and Klein (2009) **–S**	78.3	70.5	74.2	84.0	71.0	76.9	71.3	45.4	55.5
Haghighi and Klein (2009) **+S**	77.9	**74.1**	75.9	81.8	**74.3**	77.9	68.2	**51.2**	58.5
ACE2004-CULOTTA-TEST									
This work (sieve)	**80.4**	71.8	75.8	**86.3**	75.4	**80.4**	**71.6**	46.2	56.1
This work (single pass)	78.4	69.2	73.5	85.1	73.9	79.1	69.5	44.1	53.9
Haghighi and Klein (2009) **–S**	74.3	66.4	70.2	83.6	71.0	76.8	66.4	38.0	48.3
Haghighi and Klein (2009) **+S**	74.8	**77.7**	**79.6**	79.6	**78.5**	79.0	57.5	**57.6**	**57.5**
Culotta et al. (2007)	–	–	–	86.7	73.2	79.3	–	–	–
Bengston and Roth (2008)	82.7	69.9	75.8	**88.3**	**74.5**	**80.8**	55.4	63.7	59.2
MUC6-TEST									
This work (sieve)	**90.5**	68.0	77.7	**91.2**	61.2	73.2	**90.3**	53.3	67.1
This work (single pass)	89.3	65.9	75.8	90.2	58.8	71.1	89.5	50.6	64.7
Haghighi and Klein (2009) **+S**	87.2	**77.3**	**81.9**	84.7	**67.3**	**75.0**	80.5	**57.8**	**67.3**
Poon and Domingos (2008)	83.0	75.8	79.2	–	–	–	63.0	57.0	60.0
Finkel and Manning (2008) **+G**	89.7	55.1	68.3	90.9	49.7	64.3	74.1	37.1	49.5
ACE2004-NWIRE									
This work (sieve)	**83.8**	73.2	**78.1**	**87.5**	71.9	**78.9**	**79.6**	46.2	**58.4**
This work (single pass)	82.2	71.5	76.5	86.2	70.0	77.3	76.9	41.9	54.2
Haghighi and Klein (2009) **+S**	77.0	**75.9**	76.5	79.4	**74.5**	76.9	66.9	**49.2**	56.7
Poon and Domingos (2008)	71.3	70.5	70.9	–	–	–	62.6	38.9	48.0
Finkel and Manning (2008) **+G**	78.7	58.5	67.1	86.8	65.2	74.5	76.1	44.2	55.9

Table 3: Results using gold mention boundaries. Where available, we show results for a given corpus grouped in two blocks: the top block shows results of unsupervised systems and the bottom block contains supervised systems. Bold numbers indicate best results in a given block. **+/-S** indicates if the (Haghighi and Klein, 2009) system includes/excludes their semantic component. **+G** marks systems that used gold NER labels.

wikipedia articles. They first identified articles relevant to the entity mentions in the test set, and then bootstrapped from known syntactic patterns for apposition and predicate-nominatives in order to learn a database of related head pairs. They show impressive gains by using these learned pairs in coreference decisions. This type of learning using test set mentions is often described as *transductive*.

Our work instead focuses on an approach that does not require access to the dataset beforehand. We thus did not include a similar semantic component in our system, given that running a bootstrapping learner whenever a new data set is encountered is not practical and, ultimately, reduces the usability of this NLP component. However, our results show that our sieve algorithm with minimal semantic information still performs as well as the Haghighi and Klein (2009) system with semantics.

6.3 Flexible Architecture

The sieve architecture offers benefits beyond improved accuracy. Its modular design provides a flexibility for features that is not available in most supervised or unsupervised systems. The sieve allows new features to be seamlessly inserted without affecting (or even understanding) the other components. For instance, once a new high precision feature (or group of features) is inserted as its own stage, it will benefit later stages with more precise clusters, but it will not interfere with their particular algorithmic decisions. This flexibility is in sharp contrast to supervised classifiers that require their models to be retrained on labeled data, and unsupervised systems that do not offer a clear insertion point for new features. It can be difficult to fully understand how a system makes a single decision, but the sieve allows for flexible usage with minimal effort.

6.4 Error Analysis

	Pronominal	Nominal	Proper	Total
Pronominal	49 / 237	116 / 317	104 / 595	269 / 1149
Nominal	79 / 351	129 / 913	61 / 986	269 / 2250
Proper	51 / 518	15 / 730	38 / 595	104 / 1843
Total	179 / 1106	260 / 1960	203 / 2176	642 / 5242

Table 4: Number of pair-wise errors produced by the sieve after transitive closure in the MUC6-TEST corpus. Rows indicate mention types; columns are types of antecedent. Each cell shows the number of precision/recall errors for that configuration. The total number of gold links in MUC6-TEST is 11,236.

Table 4 shows the number of incorrect pair-wise links generated by our system on the MUC6-TEST corpus. The table indicates that most of our errors are for nominal mentions. For example, the combined (precision plus recall) number of errors for proper or common noun mentions is three times larger than the number of errors made for pronominal mentions. The table also highlights that most of our errors are recall errors. There are eight times more recall errors than precision errors in our output. This is a consequence of our decision to prioritize highly precise features in the sieve.

The above analysis illustrates that our next effort should focus on improving recall. In order to understand the limitations of our current system, we randomly selected 60 recall errors (20 for each mention type) and investigated their causes. Not surprisingly, the causes are unique to each type.

For proper nouns, 50% of recall errors are due to *mention lengthening*, mentions that are longer than their earlier mentions. For example, *Washington-based USAir* appears after *USAir* in the text, so our head matching components skip it because their high precision depends on disallowing new modifiers as the discourse proceeds. When the mentions were reversed (as is the usual case), they match.

The common noun recall errors are very different from proper nouns: 17 of the 20 random examples can be classified as *semantic knowledge*. These errors are roughly evenly split between recognizing categories of names (e.g., Gitano is an organization name hence it should match the nominal antecedent *the company*), and understanding hypernym relations like *settlements* and *agreements*.

Pronoun errors come in two forms. Roughly 40% of these errors are attribute mismatches involving sometimes ambiguous uses of gender and number (e.g., *she* with *Pat Carney*). Another 40% are not semantic or attribute-based, but rather simply arise due to the order in which we check potential antecedents. In all these situations, the correct links are missed because the system chooses a closer (incorrect) antecedent.

These four highlighted errors (lengthening, semantics, attributes, ordering) add up to 77% of all recall errors in the selected set. In general, each error type is particular to a specific mention type. This suggests that recall improvements can be made by focusing on one mention type without aversely affecting the others. Our sieve-based approach to coreference uniquely allows for such new models to be seamlessly inserted.

7 Conclusion

We presented a simple deterministic approach to coreference resolution that incorporates document-level information, which is typically exploited only by more complex, joint learning models. Our sieve architecture applies a battery of deterministic coreference models one at a time from highest to lowest precision, where each model builds on the previous model's cluster output. Despite its simplicity, our approach outperforms or performs comparably to the state of the art on several corpora.

An additional benefit of the sieve framework is its modularity: new features or models can be inserted in the system with limited understanding of the other features already deployed. Our code is publicly released[5] and can be used both as a stand-alone coreference system and as a platform for the development of future systems.

[5] http://nlp.stanford.edu/software/dcoref.shtml

The strong performance of our system suggests the use of sieves in other NLP tasks for which a variety of very high-precision features can be designed and non-local features can be shared; likely candidates include relation and event extraction, template slot filling, and author name deduplication.

Acknowledgments

We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of DARPA, AFRL, or the US government.

Many thanks to Jenny Finkel for writing a reimplementation of much of Haghighi and Klein (2009), which served as the starting point for the work reported here. We also thank Nicholas Rizzolo and Dan Roth for helping us replicate their experimental setup, and Heng Ji and Dekang Lin for providing their gender lexicon.

References

B. Amit and B. Baldwin. 1998. Algorithms for scoring coreference chains. In *MUC-7*.

E. Bengston and D. Roth. 2008. Understanding the value of features for coreference resolution. In *EMNLP*.

S. Bergsma and D. Lin. 2006. Bootstrapping Path-Based Pronoun Resolution. In *ACL-COLING*.

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

M. Collins and Y. Singer. 1999. Unsupervised models for named entity classification. In *EMNLP-VLC*.

A. Culotta, M. Wick, R. Hall, and A. McCallum. 2007. First-order probabilistic models for coreference resolution. In *NAACL-HLT*.

M. Elsner and E. Charniak. 2010. The same-head heuristic for coreference. In *ACL*.

J. Finkel, T. Grenager, and C. Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *ACL*.

J. Finkel and C. Manning. 2008. Enforcing transitivity in coreference resolution. In *ACL*.

B. A. Fox 1993. *Discourse structure and anaphora: written and conversational English*. Cambridge University Press.

J. Ghosh. 2003. Scalable clustering methods for data mining. *Handbook of Data Mining*, chapter 10, pages 247–277.

A. Haghighi and D. Klein. 2009. Simple coreference resolution with rich syntactic and semantic features. In *EMNLP*.

A. Haghighi and D. Klein. 2010. Coreference resolution in a modular, entity-centered model. In *HLT-NAACL*.

J.R. Hobbs. 1977. Resolving pronoun references. *Lingua*.

H. Ji and D. Lin. 2009. Gender and animacy knowledge discovery from web-scale n-grams for unsupervised person mention detection. In *PACLIC*.

L. Kertz, A. Kehler, and J. Elman. 2006. Grammatical and Coherence-Based Factors in Pronoun Interpretation. In *Proceedings of the 28th Annual Conference of the Cognitive Science Society*.

D. Klein and C. Manning. 2003. Accurate unlexicalized parsing. In *ACL*.

X. Luo. 2005. On coreference resolution performance metrics. In *HTL-EMNLP*.

H. Poon and P. Domingos. 2008. Joint unsupervised coreference resolution with Markov Logic. In *EMNLP*.

A.S. Schwartz and M.A. Hearst. 2003. A simple algorithm for identifying abbrevation definitions in biomedical text. In *Pacific Symposium on Biocomputing*.

B.F. Skinner. 1938. *The behavior of organisms: An experimental analysis*. Appleton-Century-Crofts.

V.I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2010. From baby steps to leapfrog: How “less is more” in unsupervised dependency parsing. In *NAACL*.

V. Stoyanov, N. Gilbert, C. Cardie, and E. Riloff. 2010. Conundrums in noun phrase coreference resolution: making sense of the state-of-the-art. In *ACL-IJCNLP*.

M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. 1995. A model-theoretic coreference scoring scheme. In *MUC-6*.

A Simple Domain-Independent Probabilistic Approach to Generation

Gabor Angeli
UC Berkeley
Berkeley, CA 94720
gangeli@berkeley.edu

Percy Liang
UC Berkeley
Berkeley, CA 94720
pliang@cs.berkeley.edu

Dan Klein
UC Berkeley
Berkeley, CA 94720
klein@cs.berkeley.edu

Abstract

We present a simple, robust generation system which performs content selection and surface realization in a unified, domain-independent framework. In our approach, we break up the end-to-end generation process into a sequence of local decisions, arranged hierarchically and each trained discriminatively. We deployed our system in three different domains—Robocup sportscasting, technical weather forecasts, and common weather forecasts, obtaining results comparable to state-of-the-art domain-specific systems both in terms of BLEU scores and human evaluation.

1 Introduction

In this paper, we focus on the problem of generating descriptive text given a world state represented by a set of database records. While existing generation systems can be engineered to obtain good performance on particular domains (e.g., Dale et al. (2003), Green (2006), Turner et al. (2009), Reiter et al. (2005), *inter alia*), it is often difficult to adapt them across different domains. Furthermore, *content selection* (what to say: see Barzilay and Lee (2004), Foster and White (2004), *inter alia*) and *surface realization* (how to say it: see Ratnaparkhi (2002), Wong and Mooney (2007), Chen and Mooney (2008), Lu et al. (2009), etc.) are typically handled separately. Our goal is to build a simple, flexible system which is domain-independent and performs content selection and surface realization in a unified framework.

We operate in a setting in which we are only given examples consisting of (i) a set of database records (input) and (ii) example human-generated text describing some of those records (output). We use the model of Liang et al. (2009) to automatically induce the correspondences between words in the text and the actual database records mentioned.

We break up the full generation process into a sequence of local decisions, training a log-linear classifier for each type of decision. We use a simple but expressive set of domain-independent features, where each decision is allowed to depend on the entire history of previous decisions, as in the model of Ratnaparkhi (2002). These long-range contextual dependencies turn out to be critical for accurate generation.

More specifically, our model is defined in terms of three types of decisions. The first type chooses records from the database (macro content selection)—for example, wind speed, in the case of generating weather forecasts. The second type chooses a subset of fields from a record (micro content selection)—e.g., the minimum and maximum temperature. The third type chooses a suitable template to render the content (surface realization)—e.g., *winds between* [min] *and* [max] *mph*; templates are automatically extracted from training data.

We tested our approach in three domains: ROBOCUP, for sportscasting (Chen and Mooney, 2008); SUMTIME, for technical weather forecast generation (Reiter et al., 2005); and WEATHERGOV, for common weather forecast generation (Liang et al., 2009). We performed both automatic (BLEU) and human evaluation. On WEATHERGOV, we

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 502–512,
MIT, Massachusetts, USA, 9-11 October 2010.

s:

pass(arg1=purple6, arg2=purple3)
kick(arg1=purple3)
badPass(arg1=purple3,arg2=pink9)
turnover(arg1=purple3,arg2=pink9)

w: *purple3 made a bad pass*
that was picked off by pink9

(a) ROBOCUP

s:

temperature(time=5pm-6am,min=48,mean=53,max=61)
windSpeed(time=5pm-6am,min=3,mean=6,max=11,mode=0-10)
windDir(time=5pm-6am,mode=SSW)
gust(time=5pm-6am,min=0,mean=0,max=0)
skyCover(time=5pm-9pm,mode=0-25)
skyCover(time=2am-6am,mode=75-100)
precipPotential(time=5pm-6am,min=2,mean=14,max=20)
rainChance(time=5pm-6am,mode=someChance)

w: *a 20 percent chance of showers after midnight . increasing clouds ,*
with a low around 48 southwest wind between 5 and 10 mph

(b) WEATHERGOV

s:

wind10m(time=6am,dir=SW,min=16,max=20,gust_min=0,gust_max=-)
wind10m(time=9pm,dir=SSW,min=28,max=32,gust_min=40,gust_max=-)
wind10m(time=12am,dir=-,min=24,max=28,gust_min=36,gust_max=-)

w: *sw 16 - 20 backing ssw 28 - 32 gusts 40 by mid evening easing 24 - 28 gusts 36 late evening*

(c) SUMTIME

Figure 1: Example scenarios (a scenario is a world state **s** paired with a text **w**) for each of the three domains. Each row in the world state denotes a record. Our generation task is to map a world state **s** (input) to a text **w** (output). Note that this mapping involves both content selection and surface realization.

achieved a BLEU score of 51.5 on the combined task of content selection and generation, which is more than a two-fold improvement over a model similar to that of Liang et al. (2009). On ROBOCUP and SUMTIME, we achieved results comparable to the state-of-the-art. most importantly, we obtained these results with a general-purpose approach that we believe is simpler than current state-of-the-art systems.

2 Setup and Domains

Our goal is to generate a text given a world state. The *world state*, denoted $\mathbf{s}$, is represented by a set of database records. Define $\mathcal{T}$ to be a set of record types, where each record type $t \in \mathcal{T}$ is associated with a set of fields $\text{FIELDS}(t)$. Each *record* $r \in \mathbf{s}$ has a record type $r.t \in \mathcal{T}$ and a *field value* $r.v[f]$ for each field $f \in \text{FIELDS}(t)$. The *text*, denoted $\mathbf{w}$, is represented by a sequence of tokenized words. We use the term *scenario* to denote a world state $\mathbf{s}$ paired with a text $\mathbf{w}$.

In this paper, we conducted experiments on three domains, which are detailed in the following subsections. Example scenarios for each domain are detailed in Figure 1.

2.1 ROBOCUP: Sportscasting

A world state in the ROBOCUP domain is a set of event records (*meaning representations* in the terminology of Chen and Mooney (2008)) generated by a robot soccer simulator. For example, the record pass(arg1=pink1,arg2=pink5) denotes a passing event; records of this type (pass) have two fields: arg1 (the agent) and arg2 (the recipient). As the game progresses, human commentators talk about some of the events in the game, e.g., *purple3 made a bad pass that was picked off by pink9*.

We used the dataset created by Chen and Mooney (2008), which contains 1919 scenarios from the 2001–2004 Robocup finals. Each scenario consists of a single sentence representing a fragment of a commentary on the game, paired with a set of candidate records, which were recorded within five seconds of the commentary. The records in the ROBOCUP dataset data were aligned by Chen and Mooney (2008). Each scenario contains on average $|\mathbf{s}| = 2.4$ records and 5.7 words. See Figure 1(a) for an example of a scenario. Content selection in this domain is choosing the single record to talk about, and surface realization is talking about it.

2.2 SumTime: Technical Weather Forecasts

Reiter et al. (2005) developed a generation system and created the SumTime-Meteo corpus, which consists of marine wind weather forecasts used by offshore oil rigs, generated by the output of weather simulators. More specifically, these forecasts describe various aspects of the wind at different times during the forecast period.

We used the version of the SumTime-Meteo corpus created by Belz (2008). The dataset consists of 469 scenarios, each containing on average $|\mathbf{s}| = 2.6$ records and 16.2 words. See Figure 1(c) for an example of a scenario. This task requires no content selection, only surface realization: The records are given in some fixed order and the task is to generate from each of these records in turn; of course, due to contextual dependencies, these records cannot be generated independently.

2.3 WeatherGov: Common Weather Forecasts

In the WeatherGov domain, the world state contains detailed information about a local weather forecast (e.g., temperature, rain chance, etc.). The text is a short forecast report based on this information.

We used the dataset created by Liang et al. (2009). The world state is summarized by records which aggregate measurements over selected time intervals. The dataset consists of 29,528 scenarios, each containing on average $|\mathbf{s}| = 36$ records and 28.7 words. See Figure 1(b) for an example of a scenario.

While SumTime and WeatherGov are both weather domains, there are significant differences between the two. SumTime forecasts are intended to be read by trained meteorologists, and thus the text is quite abbreviated. On the other hand, WeatherGov texts are intended to be read by the general public and thus is more English-like. Furthermore, SumTime does not require content selection, whereas content selection is a major focus of WeatherGov. Indeed, on average, only 5 of 36 records are actually mentioned in a WeatherGov scenario. Also, WeatherGov is more complex: The text is more varied, there are multiple record types, and there are about ten times as many records in each world state.

```
Generation Process
for $i = 1, 2, \ldots$ :
  **choose** a record $r_i \in \mathbf{s}$
  if $r_i = \textsc{stop}$: return
  **choose** a field set $F_i \subset \textsc{Fields}(r_i.t)$
  **choose** a template $T_i \in \textsc{Templates}(r_i.t, F_i)$
```

Figure 2: Pseudocode for the generation process. The generated text $\mathbf{w}$ is a deterministic function of the decisions.

3 The Generation Process

To model the process of generating a text $\mathbf{w}$ from a world state $\mathbf{s}$, we decompose the generation process into a sequence of local decisions. There are two aspects of this decomposition that we need to specify: (i) how the decisions are structured; and (ii) what pieces of information govern the decisions.

The decisions are structured hierarchically into three types of decisions: (i) *record decisions*, which determine which records in the world state to talk about (macro content selection); (ii) *field set decisions*, which determine which fields of those records to mention (micro content selection); and (iii) *template decisions*, which determine the actual words to use to describe the chosen fields (surface realization). Figure 2 shows the pseudocode for the generation process, while Figure 3 depicts an example of the generation process on a WeatherGov scenario.

Each of these decisions is governed by a set of feature templates (see Figure 4), which are represented as functions of the current decision and past decisions. The feature weights are learned from training data (see Section 4.3).

We chose a set of generic domain-independent feature templates, described in the sections below. These features can, in general, depend on the current decision and *all* previous decisions. For example, referring to Figure 4, **R2** features on the record choice depend on all the previous record decisions, and **R5** features depend on the most recent template decision. This is in contrast with most systems for content selection (Barzilay and Lee, 2004) and surface realization (Belz, 2008), where decisions must decompose locally according to either a graph or tree. The ability to use global features in this manner is

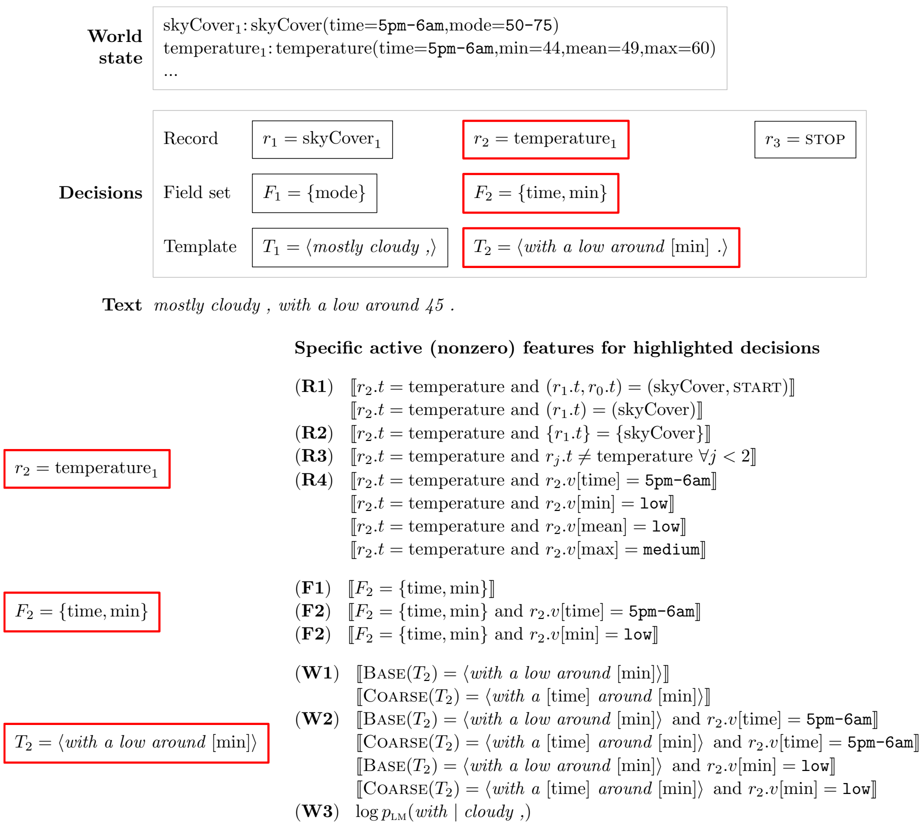

World state
skyCover$_1$: skyCover(time=5pm-6am,mode=50-75)
temperature$_1$: temperature(time=5pm-6am,min=44,mean=49,max=60)
...
Decisions
Record
r_1 = skyCover$_1$
r_2 = temperature$_1$
r_3 = STOP
Field set
$F_1 = \{\text{mode}\}$
$F_2 = \{\text{time}, \text{min}\}$
Template
$T_1 = \langle$*mostly cloudy ,*$\rangle$
$T_2 = \langle$*with a low around* [min] *.*$\rangle$
Text *mostly cloudy , with a low around 45 .*
Specific active (nonzero) features for highlighted decisions
r_2 = temperature$_1$
(R1) $\llbracket r_2.t = \text{temperature and } (r_1.t, r_0.t) = (\text{skyCover}, \text{START}) \rrbracket$
$\llbracket r_2.t = \text{temperature and } (r_1.t) = (\text{skyCover}) \rrbracket$
(R2) $\llbracket r_2.t = \text{temperature and } \{r_1.t\} = \{\text{skyCover}\} \rrbracket$
(R3) $\llbracket r_2.t = \text{temperature and } r_j.t \neq \text{temperature } \forall j < 2 \rrbracket$
(R4) $\llbracket r_2.t = \text{temperature and } r_2.v[\text{time}] = \texttt{5pm-6am} \rrbracket$
$\llbracket r_2.t = \text{temperature and } r_2.v[\text{min}] = \texttt{low} \rrbracket$
$\llbracket r_2.t = \text{temperature and } r_2.v[\text{mean}] = \texttt{low} \rrbracket$
$\llbracket r_2.t = \text{temperature and } r_2.v[\text{max}] = \texttt{medium} \rrbracket$
$F_2 = \{\text{time}, \text{min}\}$
(F1) $\llbracket F_2 = \{\text{time}, \text{min}\} \rrbracket$
(F2) $\llbracket F_2 = \{\text{time}, \text{min}\} \text{ and } r_2.v[\text{time}] = \texttt{5pm-6am} \rrbracket$
(F2) $\llbracket F_2 = \{\text{time}, \text{min}\} \text{ and } r_2.v[\text{min}] = \texttt{low} \rrbracket$
$T_2 = \langle$*with a low around* [min]$\rangle$
(W1) $\llbracket \text{BASE}(T_2) = \langle$*with a low around* [min]$\rangle \rrbracket$
$\llbracket \text{COARSE}(T_2) = \langle$*with a* [time] *around* [min]$\rangle \rrbracket$
(W2) $\llbracket \text{BASE}(T_2) = \langle$*with a low around* [min]$\rangle$ and $r_2.v[\text{time}] = \texttt{5pm-6am} \rrbracket$
$\llbracket \text{COARSE}(T_2) = \langle$*with a* [time] *around* [min]$\rangle$ and $r_2.v[\text{time}] = \texttt{5pm-6am} \rrbracket$
$\llbracket \text{BASE}(T_2) = \langle$*with a low around* [min]$\rangle$ and $r_2.v[\text{min}] = \texttt{low} \rrbracket$
$\llbracket \text{COARSE}(T_2) = \langle$*with a* [time] *around* [min]$\rangle$ and $r_2.v[\text{min}] = \texttt{low} \rrbracket$
(W3) $\log p_{\text{LM}}(\textit{with} \mid \textit{cloudy ,})$

Figure 3: The generation process on an example WEATHERGOV scenario. The figure is divided into two parts: The upper part of the figure shows the generation of text from the world state via a sequence of seven decisions (in boxes). Three of these decisions are highlighted and the features that govern these decisions are shown in the lower part of the figure. Note that different decisions in the generation process would result in different features being active (nonzero).

Feature Templates

Record	**R1**†	list of last k record types	$\llbracket r_i.t = * \text{ and } (r_{i-1}.t, \dots, r_{i-k}.t) = * \rrbracket$ for $k \in \{1, 2\}$
	R2	set of previous record types	$\llbracket r_i.t = * \text{ and } \{r_j.t : j < i\} = * \rrbracket$
	R3	record type already generated	$\llbracket r_j.t = r_i.t \text{ for some } j < i \rrbracket$
	R4	field values	$\llbracket r_i.t = * \text{ and } r_i.v[f] = * \rrbracket$ for $f \in \text{FIELDS}(r_i.t)$
	R5†	stop under language model (LM)	$\llbracket r_i.t = \text{STOP} \rrbracket \times \log p_{\text{LM}}(\text{STOP} \mid \text{previous two words generated})$
Field set	**F1**†	field set	$\llbracket F_i = * \rrbracket$
	F2	field values	$\llbracket F_i = * \text{ and } r_i.v[f] = * \rrbracket$ for $f \in F_i$
Template	**W1**†	base/coarse generation template	$\llbracket h(T_i) = * \rrbracket$ for $h \in \{\text{BASE}, \text{COARSE}\}$
	W2	field values	$\llbracket h(T_i) = * \text{ and } r_i.v[f] = * \rrbracket$ for $f \in F_i, h \in \{\text{BASE}, \text{COARSE}\}$
	W3†	first word of template under LM	$\log p_{\text{LM}}(\text{first word in } T_i \mid \text{previous two words})$

Figure 4: Feature templates that govern the record, field set, and template decisions. Each line specifies the name, informal description, and formal description of a set of features, obtained by ranging $*$ over possible values (for example, for $\llbracket r_i.t = * \rrbracket$, $*$ ranges over all record types $\mathcal{T}$). Notation: $\llbracket e \rrbracket$ returns 1 if the expression e is true and 0 if it is false. These feature templates are domain-independent; that is, they are used to create features automatically across domains. Feature templates marked with † are included in our baseline system (Section 5.2).

one of the principal advantages of our approach.

3.1 Record Decisions

Record decisions are responsible for macro content selection. Each record decision chooses a record r_i from the world state s according to features of the following types:

R1 captures the discourse coherence aspect of content selection; for example, we learn that windSpeed tends to follow windDir (but not always). **R2** captures an unordered notion of coherence—simply which sets of record types are preferable; for example, we learn that rainChance is not generated if sleetChance already was mentioned. **R3** is a coarser version of **R2**, capturing how likely it is to propose a record of a type that has already been generated. **R4** captures the important aspect of content selection that the records chosen depend on their field values;[1] for example, we learn that snowChance is not chosen unless there is snow. **R5** allows the language model to indicate whether a STOP record is appropriate; this helps prevent sentences from ending abruptly.

3.2 Field Set Decisions

Field set decisions are responsible for micro content selection, i.e., which fields of a record are mentioned. Each field set decision chooses a subset of fields F_i from the set of fields $\textsc{Fields}(r_i.t)$ of the record r_i that was just generated. These decisions are made based on two types of features:

F1 captures which sets of fields are talked about together; for example, we learn that $\{\text{mean}\}$ and $\{\text{min}, \text{max}\}$ are preferred field sets for the windSpeed record. By defining features on the entire field set, we can capture any correlation structure over the fields; in contrast, Liang et al. (2009) generates a sequence of fields in which a field can only depend on the previous one.

F2 allows the field set to be chosen based on the values of the fields, analogously to **R4**.

3.3 Template Decisions

Template decisions perform surface realization. A template is a sequence of elements, where each element is either a word (e.g., *around*) or a field (e.g., [min]). Given the record r_i and field set F_i that we are generating from, the goal is to choose a template T_i (Section 4.3.2 describes how we define the set of possible templates). The features that govern the choice of T_i are as follows:

W1 captures a priori preferences for generation templates given field sets. There are two ways to control this preference, BASE and COARSE. $\textsc{Base}(T_i)$ denotes the template T_i itself, thus allowing us to remember exactly which templates were useful. To guard against overfitting, we also use $\textsc{Coarse}(T_i)$, which maps T_i to a coarsened version of T_i, in which more words are replaced with their associated fields (see Figure 5 for an example).

W2 captures a dependence on the values of fields in the field set, and is analogous to **R4** and **F2**. Finally, **W3** contributes a language model probability, to ensure smooth transitions between templates.

After T_i has been chosen, each field in the template is replaced with a word given the corresponding field value in the world state. In particular, a word is chosen from the parameters learned in the model of Liang et al. (2009). In the example in Figure 3, the [min] field in T_2 has value 44, which is rendered to the word *45* (rounding and other noisy deviations are common in the WEATHERGOV domain).

4 Learning a Probabilistic Model

Having described all the features, we now present a conditional probabilistic model over texts $\mathbf{w}$ given world states $\mathbf{s}$ (Section 4.1). Section 4.2 describes how to use the model for generation, and Section 4.3 describes how to learn the model.

4.1 Model

Recall from Section 3 that the generation process generates $r_1, F_1, T_1, r_2, F_2, T_2, \ldots, \textsc{stop}$. To unify notation, denote this sequence of decisions as $\mathbf{d} = (d_1, \ldots, d_{|\mathbf{d}|})$.

Our probability model is defined as follows:

$$p(\mathbf{d} \mid \mathbf{s}; \theta) = \prod_{j=1}^{|\mathbf{d}|} p(d_j \mid \mathbf{d}_{<j}; \theta), \qquad (1)$$

where $\mathbf{d}_{<j} = (d_1, \ldots, d_{j-1})$ is the history of decisions and θ are the model parameters (feature weights). Note that the text $\mathbf{w}$ (the output) is a deterministic function of the decisions $\mathbf{d}$. We use the features described in Section 3 to define a log-linear model for each decision:

$$p(d_j \mid \mathbf{d}_{<j}, \mathbf{s}; \theta) = \frac{\exp\{\phi_j(d_j, \mathbf{d}_{<j}, \mathbf{s})^\top \theta\}}{\sum_{d'_j \in \mathcal{D}_j} \exp\{\phi_j(d'_j, \mathbf{d}_{<j}, \mathbf{s})^\top \theta\}}, \qquad (2)$$

where θ are all the parameters (feature weights), ϕ_j is the feature vector for the j-th decision, and $\mathcal{D}_j$ is

[1] We map a numeric field value onto one of five categories (`very-low`, `low`, `medium`, `high`, or `very-high`) based on its value with respect to the mean and standard deviation of values of that field in the training data.

the domain of the j-th decision (either records, field sets, or templates).

This chaining of log-linear models was used in Ratnaparkhi (1998) for tagging and parsing, and in Ratnaparkhi (2002) for surface realization. The ability to condition on arbitrary histories is a defining property of these models.

4.2 Using the Model for Generation

Suppose we have learned a model with parameters θ (how to obtain θ is discussed in Section 4.3). Given a world state $\mathbf{s}$, we would like to use our model to generate an output text $\mathbf{w}$ via a decision sequence $\mathbf{d}$.

In our experiments, we choose $\mathbf{d}$ by sequentially choosing the best decision in a greedy fashion (until the STOP record is generated):

$$d_j = \operatorname*{argmax}_{d'_j} p(d'_j \mid \mathbf{d}_{<j}, \mathbf{s}; \theta). \qquad (3)$$

Alternatively, instead of choosing the best decision at each point, we can sample from the distribution: $d_j \sim p(d_j \mid \mathbf{d}_{<j}, \mathbf{s}; \theta)$, which provides more diverse generated texts at the expense of a slight degradation in quality.

Both greedy search and sampling are very efficient. Another option is to try to find the Viterbi decision sequence, i.e., the one with the maximum joint probability: $\mathbf{d} = \operatorname{argmax}_{\mathbf{d}'} p(\mathbf{d}' \mid \mathbf{s}; \theta)$. However, this computation is intractable due to features depending arbitrarily on past decisions, making dynamic programming infeasible. We tried using beam search to approximate this optimization, but we actually found that beam search performed worse than greedy. Belz (2008) also found that greedy was more effective than Viterbi for their model.

4.3 Learning

Now we turn our attention to learning the parameters θ of our model. We are given a set of N scenarios $\{(\mathbf{s}^{(i)}, \mathbf{w}^{(i)})\}_{i=1}^{N}$ as training data. Note that our model is defined over the decision sequence $\mathbf{d}$ which contains information not present in $\mathbf{w}$. In Sections 4.3.1 and 4.3.2, we show how we fill in this missing information to obtain $\mathbf{d}^{(i)}$ for each training scenario i.

Assuming this missing information is filled, we end up with a standard supervised learning problem, which can be solved by maximize the (conditional) likelihood of the training data:

$$\max_{\theta \in \mathbb{R}^d} \left(\sum_{i=1}^{N} \sum_{j=1}^{|\mathbf{d}^{(i)}|} \log p(d_j^{(i)} \mid \mathbf{d}_{<j}^{(i)}; \theta) \right) - \lambda ||\theta||^2, \quad (4)$$

where $\lambda > 0$ is a regularization parameter. The objective function in (4) is optimized using the standard L-BFGS algorithm (Liu and Nocedal, 1989).

4.3.1 Latent Alignments

As mentioned previously, our training data includes only the world state $\mathbf{s}$ and generated text $\mathbf{w}$, not the full sequence of decisions $\mathbf{d}$ needed for training. Intuitively, we know *what* was generated but not *why* it was generated.

We use the model of Liang et al. (2009) to impute the decisions $\mathbf{d}$. They introduce a generative model $p(\mathbf{a}, \mathbf{w} | \mathbf{s})$, where the latent *alignment* $\mathbf{a}$ specifies (1) the sequence of records that were chosen, (2) the sequence of fields that were chosen, and (3) which words in the text were spanned by the chosen records and fields. The model is learned in an unsupervised manner using EM to produce $\mathbf{a}$ observing only $\mathbf{w}$ and $\mathbf{s}$.

An example of an alignment is given in the left part of Figure 5. This information specifies the record decisions and a set of fields for each record. Because the induced alignments can be noisy, we need to process them to obtain cleaner template decisions. This is the subject of the next section.

4.3.2 Template Extraction

Given an aligned training scenario (Figure 5), we would like to extract two types of templates.

For each record, an aligned training scenario specifies a sequence of fields and the text that is spanned by each field. We create a template by abstracting fields—that is, replacing the words spanned by a field by the field itself. We call the resulting template COARSE. The problem with using this template directly is that fields can be noisy due to errors from the unsupervised model.

Therefore, we also create a BASE template which only abstracts a subset of the fields. In particular, we define a *trigger pattern* which specifies a simple condition under which a field should be abstracted. For WEATHERGOV, we only abstract fields that

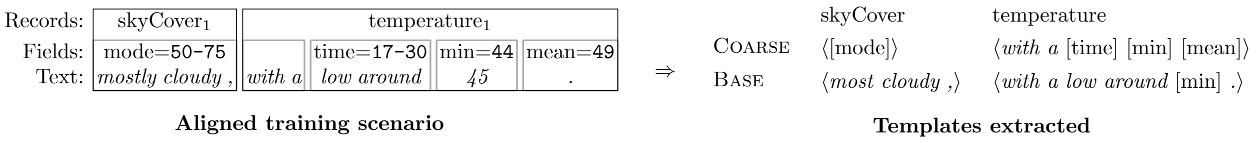


Figure 5: An example of template extraction from an imperfectly aligned training scenario. Note that these alignments are noisy (e.g., [mean] aligns to a period). Therefore, for each record (skyCover and temperature in this case), we extract two templates: (1) a COARSE template, which takes the text spanned by the record and abstracts away all fields in the scenario ([mode], [time], [min], and [mean] in the example); and (2) a BASE template, which only abstracts away fields whose spanned text matches a simple pattern (e.g., numbers in WEATHERGOV, corresponding to [min] in the example).

span numbers; for SUMTIME, fields that span numbers and wind directions; and for ROBOCUP, fields that span words starting with *purple* or *pink*.

For each record r_i, we define T_i so that $\text{BASE}(T_i)$ and $\text{COARSE}(T_i)$ are the corresponding two extracted templates. We restrict F_i to the set of abstracted fields in the COARSE template

5 Experiments

We now present an empirical evaluation of our system on our three domains—ROBOCUP, SUMTIME, and WEATHERGOV.

5.1 Evaluation Metrics

Automatic Evaluation To evaluate surface realization (or, combined content selection and surface realization), we measured the BLEU score (Papineni et al., 2002) (the precision of 4-grams with a brevity penalty) of the system-generated output with respect to the human-generated output.

To evaluate macro content selection, we measured the F_1 score (the harmonic mean of precision and recall) of the set of records chosen with respect to the human-annotated set of records.

Human Evaluation We conducted a human evaluation using Amazon Mechanical Turk. For each domain, we chose 100 scenarios randomly from the test set. We ran each system under consideration on each of these scenarios, and presented each resulting output to 10 evaluators.[2] Evaluators were given instructions to rank an output on the basis of English fluency and semantic correctness on the following scale:

Score	English Fluency	Semantic Correctness
5	Flawless	Perfect
4	Good	Near Perfect
3	Non-native	Minor Errors
2	Disfluent	Major Errors
1	Gibberish	Completely Wrong

Evaluators were also given additional domain-specific information: (1) the background of the domain (e.g., that SUMTIME reports are technical weather reports); (2) general properties of the desired output (e.g., that SUMTIME texts should mention every record whereas WEATHERGOV texts need not); and (3) peculiarities of the text (e.g., the suffix *ly* in SUMTIME should exist as a separate token from its stem, or that *pink goalie* and *pink1* have the same meaning in ROBOCUP).

5.2 Systems

We evaluated the following systems on our three domains:

- HUMAN is the human-generated output.
- OURSYSTEM uses all the features in Figure 4 and is trained according to Section 4.3.
- BASELINE is OURSYSTEM using a subset of the features (those marked with † in Figure 4). In contrast to OURSYSTEM, the included features only depend on a local context of decisions in a manner similar to the generative model of Liang et al. (2009) and the *p*CRU-greedy system of Belz (2008). BASELINE also excludes features that depend on values of the world state.
- The existing state-of-the-art domain-specific system for each domain.

5.3 ROBOCUP Results

Following the evaluation methodology of Chen and Mooney (2008), we trained our system on three

[2] To minimize bias, we evaluated all the systems at once, randomly shuffling the outputs of the systems. The evaluators were not necessarily the same 10 evaluators.

System	F$_1$	BLEU*	English Fluency	Semantic Correctness
BASELINE	78.7	24.8	4.28 ± 0.78	4.15 ± 1.14
OURSYSTEM	79.9	28.8	4.34 ± 0.69	4.17 ± 1.21
WASPER-GEN	72.0	28.7	4.43 ± 0.76	4.27 ± 1.15
HUMAN	—	—	4.43 ± 0.69	4.30 ± 1.07

Table 1: ROBOCUP results. WASPER-GEN is described in Chen and Mooney (2008). The BLEU is reported on systems that use fixed human-annotated records (in other words, we evaluate surface realization given perfect content selection).

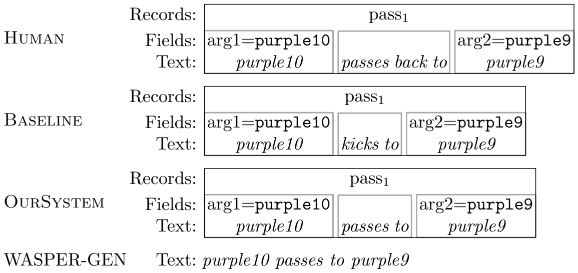

HUMAN
Records: pass$_1$
Fields: arg1=purple10 | arg2=purple9
Text: purple10 | passes back to | purple9
BASELINE
Records: pass$_1$
Fields: arg1=purple10 | arg2=purple9
Text: purple10 | kicks to | purple9
OURSYSTEM
Records: pass$_1$
Fields: arg1=purple10 | arg2=purple9
Text: purple10 | passes to | purple9
WASPER-GEN
Text: purple10 passes to purple9

Figure 6: Outputs of systems on an example ROBOCUP scenario. There are some minor differences between the outputs. Recall that OURSYSTEM differs from BASELINE mostly in the addition of feature **W2**, which captures dependencies between field values (e.g., `purple10`) and the template chosen (e.g., [arg1] *passes to* [arg2]). This allows us to capture value-dependent preferences for different realizations (e.g., *passes to* over *kicks to*). Also, HUMAN uses *passes back to*, but this word choice requires knowledge of passing records in previous scenarios, which none of the systems have access to. It would natural, however, to add features that would capture these longer-range dependencies in our framework.

Robocup games and tested on the fourth, averaging over the four train/test splits. We report the average test accuracy weighted by the number of scenarios in a game. First, we evaluated macro content selection. Table 1 shows that OURSYSTEM significantly outperforms BASELINE and WASPER-GEN on F$_1$.

To compare with Chen and Mooney (2008) on surface realization, we fixed each system's record decisions to the ones given by the annotated data and enforced that all the fields of that record are chosen. Table 1 shows that OURSYSTEM significantly outperforms BASELINE and is comparable to WASPER-GEN on BLEU. On human evaluation, OURSYSTEM outperforms BASELINE, but WASPER-GEN outperforms OURSYSTEM. See Figure 6 for example outputs from the various systems.

	BLEU	English Fluency	Semantic Correctness
BASELINE	32.9	4.23 ± 0.71	4.26 ± 0.85
OURSYSTEM	55.1	4.25 ± 0.69	4.27 ± 0.82
OURSYSTEM-CUSTOM	62.3	4.12 ± 0.78	4.33 ± 0.91
*p*CRU-greedy	63.6	4.18 ± 0.71	4.49 ± 0.73
SUMTIME-Hybrid	52.7	—	—
HUMAN	—	4.09 ± 0.83	4.37 ± 0.87

Table 2: SUMTIME results. The SUMTIME-Hybrid system is described in (Reiter et al., 2005); *p*CRU-greedy, in (Belz, 2008).

5.4 SUMTIME Results

The SUMTIME task only requires micro content selection and surface realization because the sequence of records to be generated is fixed; only these aspects are evaluated. Following the methodology of Belz (2008), we used five-fold cross validation.

We found that using the unsupervised model of Liang et al. (2009) to automatically produce aligned training scenarios (Section 4.3.1) was less effective than it was in the other two domains due to two factors: (i) there are fewer training examples in SUMTIME and unsupervised learning typically works better with a large amount of data; and (ii) the alignment model does not exploit the temporal structure in the SUMTIME world state. Therefore, we used a small set of simple regular expressions to produce aligned training scenarios.

Table 2 shows that OURSYSTEM significantly outperforms BASELINE as well as SUMTIME-Hybrid, a hand-crafted system, on BLEU. Note that OURSYSTEM is domain-independent and has not been specifically tuned to SUMTIME. However, OURSYSTEM is outperformed by the state-of-the-art statistical system *p*CRU-greedy.

Custom Features One of the advantages of our feature-based approach is that it is straightforward to incorporate domain-specific features to capture specific properties of a domain. To this end, we define the following set of feature templates in place of our generic feature templates from Figure 4:

- **F1′**: Value of time
- **F2′**: Existence of gusts/wind direction/wind speeds
- **W1′**: Change in wind direction (clockwise, counterclockwise, or none)

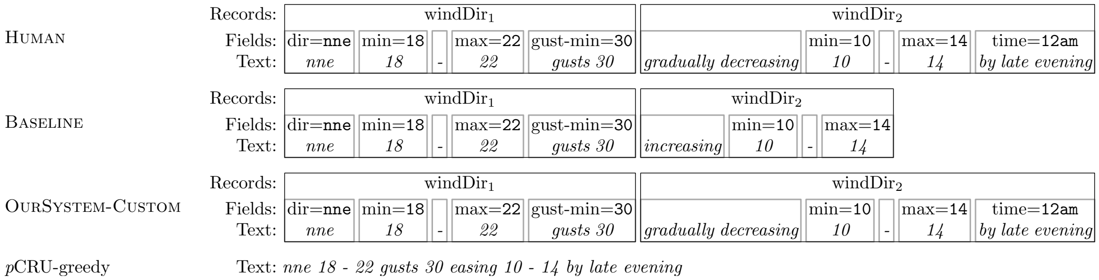

HUMAN
Records: windDir$_1$ windDir$_2$
Fields: dir=nne min=18 max=22 gust-min=30 min=10 max=14 time=12am
Text: nne 18 - 22 gusts 30 gradually decreasing 10 - 14 by late evening
BASELINE
Records: windDir$_1$ windDir$_2$
Fields: dir=nne min=18 max=22 gust-min=30 min=10 max=14
Text: nne 18 - 22 gusts 30 increasing 10 - 14
OURSYSTEM-CUSTOM
Records: windDir$_1$ windDir$_2$
Fields: dir=nne min=18 max=22 gust-min=30 min=10 max=14 time=12am
Text: nne 18 - 22 gusts 30 gradually decreasing 10 - 14 by late evening
pCRU-greedy
Text: nne 18 - 22 gusts 30 easing 10 - 14 by late evening

Figure 7: Outputs of systems on an example SUMTIME scenario. Two notable differences between OURSYSTEM-CUSTOM and BASELINE arise due to OURSYSTEM-CUSTOM's value-dependent features. For example, OURSYSTEM-CUSTOM can choose whether to include the time field (windDir$_2$) or not (windDir$_1$), depending on the value of the time (**F1$'$**), thereby improving content selection. OURSYSTEM-CUSTOM also improves surface realization, choosing *gradually decreasing* over BASELINE's *increasing*. Interestingly, this improvement comes from the joint effort of two features: **W2$'$** prefers *decreasing* over *increasing* in this case, and **W5$'$** adds the modifier *gradually*. An important strength of log-linear models is the ability to combine soft preferences from many features.

- **W2$'$**: Change in wind speed
- **W3$'$**: Change in wind direction and speed
- **W4$'$**: Existence of gust min and/or max
- **W5$'$**: Time elapsed since last record
- **W6$'$**: Whether wind is a cardinal direction (N, E, S, W)

The resulting system, which we call OURSYSTEM-CUSTOM, obtains a BLEU score which is comparable to *p*CRU-greedy.

An important aspect of our system that it is flexible and quick to deploy. According to Belz (2008), SUMTIME-Hybrid took twelve person-months to build, while *p*CRU-greedy took one month. Having developed OURSYSTEM in a domain-independent way, we only needed to do simple reformatting upon receiving the SUMTIME data. Furthermore, it took only a few days to develop the custom features above to create OURSYSTEM-CUSTOM, which has BLEU performance comparable to the state-of-the-art *p*CRU-greedy system.

We also conducted human evaluations on the four systems shown in Table 2. Note that this evaluation is rather difficult for Mechanical Turkers since SUMTIME texts are rather technical compared to those in other domains. Interestingly, all systems outperform HUMAN on English fluency; this result corroborates the findings of Belz (2008). On semantic correctness, all systems perform comparably to HUMAN, except *p*CRU-greedy, which performs slightly better. See Figure 7 for a comparison of the outputs generated by the various systems.

	F$_1$	BLEU*	English Fluency	Semantic Correctness
BASELINE	22.1	22.2	4.07 ± 0.59	3.41 ± 1.16
OURSYSTEM	65.4	51.5	4.12 ± 0.74	4.22 ± 0.89
HUMAN	—	—	4.14 ± 0.71	3.85 ± 0.99

Table 3: WEATHERGOV results. The BLEU score is on joint content selection and surface realization and is modified to not penalize numeric deviations of at most 5.

5.5 WEATHERGOV Results

We evaluate the WEATHERGOV corpus on the joint task of content selection and surface realization. We split our corpus into 25,000 scenarios for training, 1,000 for development, and 3,528 for testing. In WEATHERGOV, numeric field values are often rounded or noisily perturbed, so it is difficult to generate precisely matching numbers. Therefore, we used a modified BLEU score where numbers differing by at most five are treated as equal. Furthermore, WEATHERGOV is evaluated on the joint content selection and surface realization task, unlike ROBOCUP, where content selection and surface realization were treated separately, and SUMTIME, where content selection was not applicable.

Table 3 shows the results. We see that OURSYSTEM substantially outperforms BASELINE, especially on BLEU score and semantic correctness. This difference shows that taking non-local context into account is very important in this domain. This result is not surprising, since WEATHERGOV is the most complicated of the three domains, and this complexity is exactly where non-locality is neces-

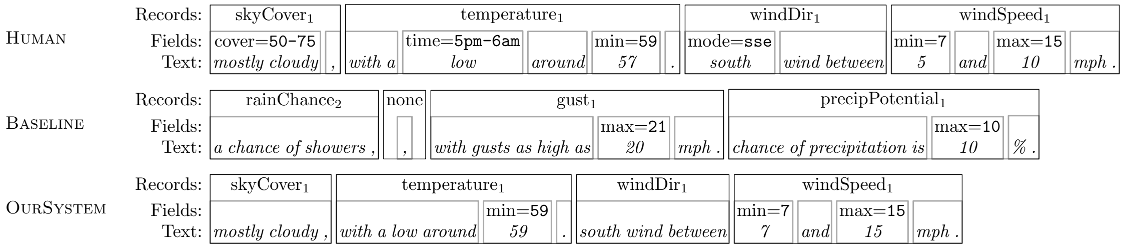

HUMAN
Records: skyCover₁ | temperature₁ | windDir₁ | windSpeed₁
Fields: cover=50-75 | time=5pm-6am | min=59 | mode=sse | min=7 | max=15
Text: mostly cloudy , with a low around 57 . south wind between 5 and 10 mph .
BASELINE
Records: rainChance₂ | none | gust₁ | precipPotential₁
Fields: max=21 | max=10
Text: a chance of showers , , with gusts as high as 20 mph . chance of precipitation is 10 % .
OURSYSTEM
Records: skyCover₁ | temperature₁ | windDir₁ | windSpeed₁
Fields: min=59 | min=7 | max=15
Text: mostly cloudy , with a low around 59 . south wind between 7 and 15 mph .

Figure 8: Outputs of systems on an example WEATHERGOV scenario. Most of the gains of OURSYSTEM over BASELINE come from improved content selection. For example, BASELINE chooses rainChance because it happens to be the most common first record type in the training data. However, since OURSYSTEM has features that depend on the value of rainChance (`noChance` in this case), it has learned to disprefer talking about rain when there is no rain. Also, OURSYSTEM has additional features on the entire history of chosen records, which enables it to choose a better sequence of records.

sary. Interestingly, OURSYSTEM even outperforms HUMAN on semantic correctness, perhaps due to generating more straightforward renderings of the world state. Figure 8 describes example outputs for each system.

6 Related Work

There has been a fair amount of work both on content selection and surface realization. In content selection, Barzilay and Lee (2004) use an approach based on local classification with edge-wise scores between local decisions. Our model, on the other hand, can capture higher-order constraints to enforce global coherence.

Liang et al. (2009) introduces a generative model of the text given the world state, and in some ways is similar in spirit to our model. Although that model is capable of generation in principle, it was designed for unsupervised induction of hidden alignments (which is exactly what we use it for). Even if combined with a language model, generated text was much worse than our baseline.

The prominent approach for surface realization is rendering the text from a grammar. Wong and Mooney (2007) and Chen and Mooney (2008) use synchronous grammars that map a logical form, represented as a tree, into a parse of the text. Soricut and Marcu (2006) uses tree structures called WIDL-expressions (the acronym corresponds to four operations akin to the rewrite rules of a grammar) to represent the realization process, and, like our approach, operates in a log-linear framework. Belz (2008) and Belz and Kow (2009) also perform surface realization from a PCFG-like grammar. Lu et al. (2009) uses a conditional random field model over trees. Other authors have performed surface realization using various grammar formalisms, for instance CCG (White et al., 2007), HPSG (Nakanishi et al., 2005), and LFG (Cahill and van Genabith, 2006).

In each of the above cases, the decomposable structure of the tree/grammar enables tractability. However, we saw that it was important to include features that captured long-range dependencies. Our model is also similar in spirit to Ratnaparkhi (2002) in the use of non-local features, but we operate at three levels of hierarchy to include both content selection and surface realization.

One issue that arises with long-range dependencies is the lack of efficient algorithms for finding the optimal text. Koller and Striegnitz (2002) perform surface realization of a flat semantics, which is NP-hard, so they recast the problem as non-projective dependency parsing. Ratnaparkhi (2002) uses beam search to find an approximate solution. We found that a greedy approach obtained better results than beam search; Belz (2008) found greedy approaches to be effective as well.

7 Conclusion

We have developed a simple yet powerful generation system that combines both content selection and surface realization in a domain independent way. Despite our approach being domain-independent, we were able to obtain performance comparable to the state-of-the-art across three domains. Additionally, the feature-based design of our approach makes it easy to incorporate domain-specific knowledge to increase performance even further.

References

R. Barzilay and L. Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*.

A. Belz and E. Kow. 2009. System building cost vs. output quality in data-to-text generation. In *European Workshop on Natural Language Generation*, pages 16–24.

A. Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, 14(4):1–26.

Aoife Cahill and Josef van Genabith. 2006. Robust pcfg-based generation using automatically acquired LFG approximations. In *Association for Computational Linguistics (ACL)*, pages 1033–1040, Morristown, NJ, USA. Association for Computational Linguistics.

D. L. Chen and R. J. Mooney. 2008. Learning to sportscast: A test of grounded language acquisition. In *International Conference on Machine Learning (ICML)*, pages 128–135.

R. Dale, S. Geldof, and J. Prost. 2003. Coral: using natural language generation for navigational assistance. In *Australasian computer science conference*, pages 35–44.

M. E. Foster and M. White. 2004. Techniques for text planning with XSLT. In *Workshop on NLP and XML: RDF/RDFS and OWL in Language Technology*, pages 1–8.

N. Green. 2006. Generation of biomedical arguments for lay readers. In *International Natural Language Generation Conference*, pages 114–121.

A. Koller and K. Striegnitz. 2002. Generation as dependency parsing. In *Association for Computational Linguistics (ACL)*, pages 17–24.

P. Liang, M. I. Jordan, and D. Klein. 2009. Learning semantic correspondences with less supervision. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

D. C. Liu and J. Nocedal. 1989. On the limited memory method for large scale optimization. *Mathematical Programming B*, 45(3):503–528.

W. Lu, H. T. Ng, and W. S. Lee. 2009. Natural language generation with tree conditional random fields. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 400–409.

Hiroko Nakanishi, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic models for disambiguation of an HPSG-based chart generator. In *Parsing '05: Proceedings of the Ninth International Workshop on Parsing Technology*, pages 93–102, Morristown, NJ, USA. Association for Computational Linguistics.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Association for Computational Linguistics (ACL)*.

A. Ratnaparkhi. 1998. *Maximum entropy models for natural language ambiguity resolution*. Ph.D. thesis, University of Pennsylvania.

A. Ratnaparkhi. 2002. Trainable approaches to surface natural language generation and their application to conversational dialog systems. *Computer, Speech & Language*, 16:435–455.

E. Reiter, S. Sripada, J. Hunter, J. Yu, and I. Davy. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, 167:137–169.

R. Soricut and D. Marcu. 2006. Stochastic language generation using WIDL-expressions and its application in machine translation and summarization. In *Association for Computational Linguistics (ACL)*, pages 1105–1112.

R. Turner, Y. Sripada, and E. Reiter. 2009. Generating approximate geographic descriptions. In *European Workshop on Natural Language Generation*, pages 42–49.

Michael White, Rajakrishnan Rajkumar, and Scott Martin. 2007. Towards broad coverage surface realization with CCG. In *In Proceedings of the Workshop on Using Corpora for NLG: Language Generation and Machine Translation (UCNLG+MT)*.

Y. W. Wong and R. J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Association for Computational Linguistics (ACL)*, pages 960–967.

Title Generation with Quasi-Synchronous Grammar

Kristian Woodsend, Yansong Feng and Mirella Lapata
School of Informatics, University of Edinburgh
Edinburgh EH8 9AB, United Kingdom
k.woodsend@ed.ac.uk, Y.Feng-4@sms.ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

The task of selecting information and rendering it appropriately appears in multiple contexts in summarization. In this paper we present a model that simultaneously optimizes selection and rendering preferences. The model operates over a phrase-based representation of the source document which we obtain by merging PCFG parse trees and dependency graphs. Selection preferences for individual phrases are learned discriminatively, while a quasi-synchronous grammar (Smith and Eisner, 2006) captures rendering preferences such as paraphrases and compressions. Based on an integer linear programming formulation, the model learns to generate summaries that satisfy both types of preferences, while ensuring that length, topic coverage and grammar constraints are met. Experiments on headline and image caption generation show that our method obtains state-of-the-art performance using essentially the same model for both tasks without any major modifications.

1 Introduction

Summarization is the process of condensing a source text into a shorter version while preserving its information content. Humans summarize on a daily basis and effortlessly, yet the automatic production of high-quality summaries remains a challenge.

Most work today focuses on *extractive summarization*, where a summary is created by identifying and subsequently concatenating the most important sentences in a document. The advantage of this approach is that it does not require a great deal of linguistic analysis to generate grammatical sentences, assuming the source document was well written. Unfortunately, extracts generated this way are often documents of low readability and text quality, and contain much redundant information. The conciseness can be improved when sentence extraction is interfaced with *sentence compression*, where words and clauses are deleted based on rules typically operating over parsed input (Jing, 2000; Daumé III and Marcu, 2002; Lin, 2003; Daumé III, 2006; Zajic et al., 2007; Martins and Smith, 2009).

An alternative *abstractive* or "bottom-up" approach involves identifying high-interest words and phrases in the source text, and combining them into new sentences guided by a language model (Banko et al., 2000; Soricut and Marcu, 2007). This approach has the potential to work well, breaking out of the single-sentence paradigm. Unfortunately, the resulting summaries are not always coherent — individual constituent phrases are often combined without any semantic constraints — or grammatical beyond the *n*-gram horizon imposed by the language model.

Constituent deletion and recombination are merely two of the many rewrite operations professional editors and abstractors employ when creating summaries (Jing, 2002). Additional operations include truncating sentences, aggregating them, and paraphrasing at word or syntax level. Furthermore, professionals write summaries in a task-specific style. News headlines for example are typically short (three to six words), written in the present tense and active voice, and often leave out forms of the verb *be*. There are also different ways of writing a headline either *directly* by stating what the docu-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 513–523,
MIT, Massachusetts, USA, 9-11 October 2010.

ment is about or *indirectly* by raising a question in the reader's mind, which the document answers.

The automatic generation of summaries similar to those produced by human abstractors is challenging because of the many constraints imposed by the task: the summary must be maximally informative and minimally redundant, grammatical, coherent, adhere to a pre-specified length and stylistic conventions. Importantly, these constraints are conflicting; the deletion of certain phrases may avoid redundancy but result in ungrammatical output and information loss.

In this paper we propose a model for summarization that attempts to capture and optimize these constraints jointly. We learn both how to select the most important information (the content), and how to render it appropriately (the style). Selection preferences are learned discriminatively, while a quasi-synchronous grammar (QG, Smith and Eisner 2006) captures rendering preferences such as paraphrases and compressions. The entire solution space of possible extractions and QG-generated paraphrases is searched efficiently through use of integer linear programming. The ILP framework allows us to model naturally as constraints, additional requirements such as sentence length, overall summary length, topic coverage and, importantly, grammaticality.

We argue that QG is attractive for describing rewrite operations common in summarization. Rather than assuming a strictly synchronous structure over the source and target sentences, QG identifies a "sloppy" alignment of parse trees assuming that the target tree is in some way "inspired by" the source tree. A key insight in our approach is to formulate the summarization problem at the *phrase* level: both QG rules and information extraction operate over individual phrases rather than (as is the norm) sentences. At this smaller unit level, QG rules become more widely applicable and compression falls naturally because only phrases deemed important should appear in the summary.

We evaluate the proposed model on headline generation and the related task of image caption generation. However, there is nothing inherent in our formulation that is specific to those two tasks; it is possible for the model to generate longer or shorter summaries, for a single or multiple documents. Experimental results show that our method obtains state-of-the-art performance, both in terms of grammaticality and informativeness for both tasks using the same summarization model.

2 Related work

Much effort in automatic summarization has been devoted to sentence extraction which is often formalized as a classification task (Kupiec et al., 1995). Given appropriately annotated training data, a binary classifier learns to predict for each document sentence if it is worth extracting. A few previous approaches have attempted to interface sentence compression with summarization. A straightforward way to achieve this is by adopting a two-stage architecture (e.g., Lin 2003) where the sentences are first extracted and then compressed or the other way round.

Other work implements a *joint* model where words are deleted and sentences selected from a document simultaneously (Daumé III and Marcu, 2002; Martins and Smith, 2009; Woodsend and Lapata, 2010). ILP models have also been developed for sentence rather than document compression (Clarke and Lapata, 2008). Dras (1999) discusses the application of ILP to reluctant paraphrasing, i.e., the task of choosing between paraphrases while conforming to length, readability, or style constraints. Again, the aim is to rewrite text without, however, content selection. Rewrite operations other than deletion tend to be hand-crafted and domain specific (Jing and McKeown, 2000). Notable exceptions are Cohn and Lapata (2008) and Zhao et al. (2009) who present a model that can both compress and paraphrase individual sentences without however generating document-level summaries.

Headline generation is a well-studied task within single-document summarization, due to its prominence in the DUC-03 and DUC-04 evaluation competitions.[1] Many approaches identify the most informative sentence in a given document (typically the first sentence for the news genre) and subsequently apply a form of sentence compression such that the headline meets some length requirement (Dorr

[1] Approaches to headline generation are too numerous to list in detail; see the proceedings of DUC-03 and DUC-04 for an overview.

et al., 2003). The compressed sentence may also be "padded" with important content words or phrases to ensure that the topic of the document is covered (Zajic et al., 2004). Other work generates headlines in a bottom-up fashion starting from important, individual words and phrases, that are glued together to create a fluent sentence. For example, Banko et al. (2000) draw inspiration from Machine Translation and generate headlines using statistical models for content selection and sentence realization.

Relatively little work has focused on caption generation, a task related to headline generation. The aim here is to create a short, title-like description of an image embedded in a news article. Like headlines, captions have to be short and informative. In addition, a good caption must clearly identify the subject of the picture and establish its relevance to the article. Feng and Lapata (2010a) develop extractive and abstractive caption generation models that operate over the output of a probabilistic image annotation model that preprocesses the pictures and suggests keywords to describe their content. Their best model is an extension of Banko et al.'s (2000) word-based model for headline generation to phrases.

Our own work develops an ILP-based summarization model with rewrite operations that are not limited to deletion, are defined over phrases, and encoded in quasi-synchronous grammar. The QG formalism has been previously applied to parser adaptation and projection (Smith and Eisner, 2009), paraphrase identification (Das and Smith, 2009), and question answering (Wang et al., 2007); however the use of QG in summarization is novel to our knowledge. Unlike most synchronous grammar formalisms, QG does not posit a strict isomorphism between a source sentence and its target translation; it only loosely links the syntactic structure of the two, and is therefore well suited to describing the relationship between a document and its abstract. We propose an ILP formulation which not only allows to efficiently search through the space of many QG rules but also to incorporate constraints relating to content, style, and the task at hand.

3 Modeling

There are three components to our model. Content selection is performed discriminatively; an SVM learns which information in the source document should be in the summary, and gives a real-valued *salience score* for each phrase. QG rules are used to generate compressions and paraphrases of the source sentences. An ILP model combines the output of these two components into an output summary, while optimizing content selection and surface realization preferences jointly.

3.1 Document Representation

Our model operates on documents annotated with syntactic information which we obtain by parsing every sentence twice, once with a phrase structure parser and once with a dependency parser. The output from the two representations is combined into a single data structure, by mapping the dependencies to the edges of the phrase structure tree. The procedure is described in detail in Woodsend and Lapata (2010). However, we do not merge the leaf nodes into phrases here, but keep the full tree structure, as we will apply compression to phrases through the QG. In our experiments, we obtain this combined representation from the output of the Stanford parser (Klein and Manning, 2003) but any other broadly similar parser could be used instead.

3.2 Quasi-synchronous grammar

Given an input sentence S1 or its parse tree T1, the QG constructs a monolingual grammar for parsing, or generating, the possible translation (or here, paraphrase) trees T2. A grammar node in the target tree T2 is modeled on a subset of nodes in the source tree, with a rather loose alignment between the trees.

In our approach, the process of learning the grammar is unsupervised. Each sentence of the source document is compared to each sentence in the target document — headline or caption, depending on the task. Using the combined PCFG-dependency tree representation described above, we build up a list of leaf node alignments based on lexical identity, after stemming and removing stop words. We align direct parent nodes where more than one child node aligns. A grammar rule is created if the all the nodes in the target tree can be explained using nodes from the

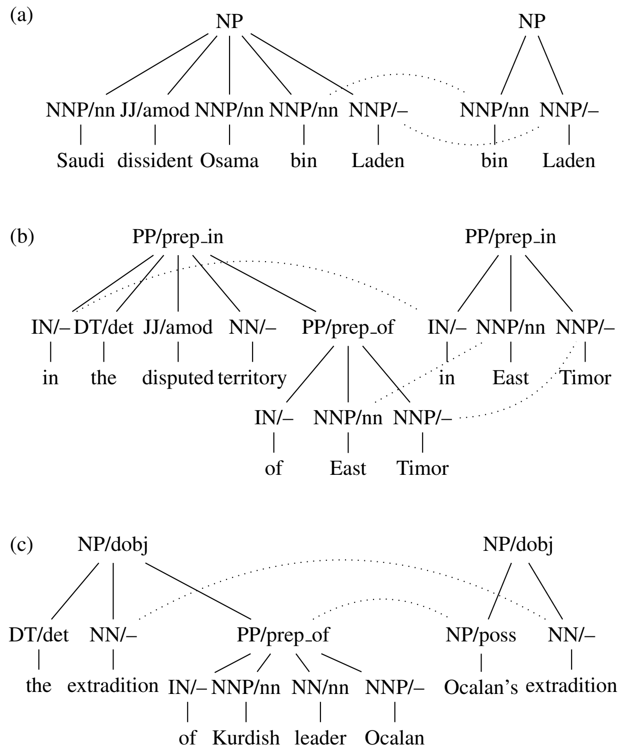


Figure 1: Examples of QG alignments between source node (left) and target node (right). (a) alignment of child nodes, involving compression through deletion; (b) rewriting involving child and grandchild nodes; (c) reordering of child nodes (with further compression through applying other QG rules on children). Nodes bear phrase and dependency labels. Dotted lines show alignments in the grammar between source and target child nodes. Examples are taken from the QG rules discovered in the DUC-03 data set of headlines.

source; this helps to improve the quality in what is inherently a noisy process. Finally, QG rules are created from aligned nodes above the leaf node level, recording the phrase and dependency label of nodes, and the alignment of child nodes.

Unlike previous work involving QG which has used dependency graphs exclusively (e.g., Wang et al. 2007; Das and Smith 2009), our approach operates over a combined PCFG-dependency representation. As a result, some configurations in Smith and Eisner (2006) are not so relevant here — instead, we found that deletions, reorderings, flattening of nodes, and the addition of text elements were important operations for the grammar.

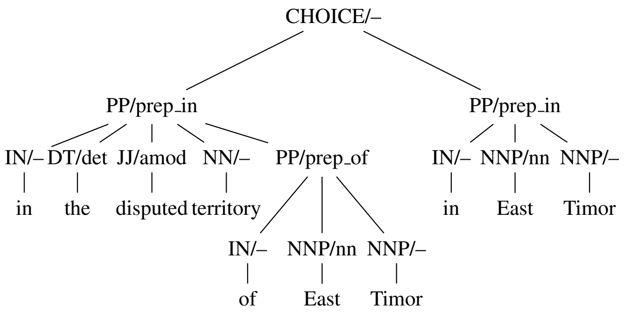


Figure 2: Alternative paraphrases are represented as a CHOICE sub-tree.

Figure 1 shows some example alignments that are captured by the QG, with the source node on the left and the target node on the right. Leaf nodes have their original text, while other nodes have a combined phrase and dependency label that they obtain in the merged representation described in Section 3.1 above (e.g., NP/dobj is a noun phrase and a direct object, NNP/nn is a proper noun and a nominal modifier, whereas NN/– is a head noun). Alignments between the children are shown by dotted lines. In Figure 1(a), some child nodes are aligned while others are not present in the target tree. This type of rule is common in our training data, and typically arises from the compression of names in noun phrases. Another frequent compression, shown in Figure 1(b), is flattening the tree structure by incorporating grand-child elements at the child level. Figure 1(c) shows a rule involving the reordering of child nodes, and where additional rules are applied recursively to achieve further compression and a transformation in the phrase constituency.

Paraphrases are created from source sentence parse trees by applying suitable rules recursively. Suitable rules have matching structure in terms of phrase and dependency label, for both the parent and child nodes. Additionally, the proposed paraphrase sub-tree must be suitable for the target tree being created (i.e., the root node of the paraphrase must match the phrase and dependency label of the corresponding node in the target tree). Where more than one paraphrase is possible, the alternatives are incorporated into the target parse tree under a *CHOICE* node, as is shown in Figure 2. Note that unlike previous QG approaches, we do not use the probability model proposed by Smith and Eisner (2006); instead the QG is used to represent rewrite operations, and we simply record a frequency count for how often each rule is encountered in the training data.

3.3 ILP model

The objective of our model is to create the most informative text possible, subject to constraints which can be tailored to the specific task. These relate to sentence length, overall summary length, the inclusion of specific topics, and grammaticality. These constraints are *global* in their scope, and cannot be adequately satisfied by optimizing each one of them individually. Our approach therefore uses an ILP formulation which will provide a globally optimal solution, and which can be efficiently solved using standard optimization tools. Specifically, the model selects phrases and paraphrases from which to form the output sentence. Here, we focus on a single sentence as this is most appropriate for title generation. However, multi-sentence output can be easily generated by setting a summary length constraint. The model operates over the merged phrase structure trees described in Section 3.1, augmented with paraphrase choice nodes such as shown in Figure 2 rather than raw text.

Let $\mathcal{S}$ be the set of sentences in a document, $\mathcal{P}$ be the set of phrases, and $\mathcal{P}_s \subset \mathcal{P}$ be the set of phrases in each sentence $s \in \mathcal{S}$. Let the sets $\mathcal{D}_i \subset \mathcal{P}, \forall i \in \mathcal{P}$ capture the phrase dependency information for each phrase i, where each set $\mathcal{D}_i$ contains the phrases that depend on the presence of i. In a similar fashion, $\mathcal{C} \subset \mathcal{P}$ is the set of choice nodes throughout the document, which represent nodes in the tree where more than one QG rule can be applied; $\mathcal{C}_i \subset \mathcal{P}, i \in \mathcal{C}$ are the sets of phrases that are direct children of each choice node, in other words they are the individual alternative paraphrases. Let l_i be the length of each phrase i, in tokens.

For caption generation, the model has as additional input a list of tags (keywords drawn from the source document) that correspond to the image, and we refer to this set of tags as $\mathcal{T}$. $\mathcal{P}_t \subset \mathcal{P}$ is the set of phrases containing the tag $t \in \mathcal{T}$. We use the probabilistic image annotation model of Feng and Lapata (2010a) to generate the list of keywords. The latter highlight the objects depicted in the image and should be in all likelihood included in the caption.

The model is cast as an integer linear program:

$$\begin{aligned}
\max_{x} \quad & \sum_{i \in \mathcal{P}} (f_i + \lambda g_i) x_i && && \text{(1a)}\\
\text{s.t.} \quad & \sum_{i \in \mathcal{P}} l_i x_i \leq L_{max} && && \text{(1b)}\\
& \sum_{i \in \mathcal{P}} l_i x_i \geq L_{min} && && \text{(1c)}\\
& \sum_{i \in \mathcal{P}_t, t \in \mathcal{T}} x_i \geq T_{min} && && \text{(1d)}\\
& x_j \rightarrow x_i && \forall i \in \mathcal{P}, j \in \mathcal{D}_i && \text{(1e)}\\
& \sum_{j \in \mathcal{C}_i} x_j = x_i && \forall i \in \mathcal{C}, j \in \mathcal{C}_i && \text{(1f)}\\
& x_i \rightarrow y_s && \forall s \in \mathcal{S}, i \in \mathcal{P}_s && \text{(1g)}\\
& \sum_{s \in \mathcal{S}} y_s \leq N_S && && \text{(1h)}\\
& x_i \in \{0,1\} && \forall i \in \mathcal{P} && \text{(1i)}\\
& y_s \in \{0,1\} && \forall s \in \mathcal{S}. && \text{(1j)}
\end{aligned}$$

A vector of binary variables $x \in \{0,1\}^{|\mathcal{P}|}$ indicates if each phrase is to be part of the output. The vector of auxiliary binary variables $y \in \{0,1\}^{|\mathcal{S}|}$ indicates from which sentences the chosen phrases come, see Equation (1g).

Our objective function (1a) is the weighted sum of two components for each phrase: a salience score, and a measure of how frequently the QG rule was seen in the training data. Let f_i denote the salience score for phrase i, determined by the machine learning algorithm. We apply a paraphrase penalty g_i to each phrase,

$$g_i = \log\left(\frac{n_r}{N_r}\right),$$

where n_r is a count of the number of times this particular QG rule r was seen in the training data, and N_r is the number of times all suitable rules for this phrase node were seen. If no suitable rules exist, we set $g_i = 0$. The intuition here is that common paraphrases should be more trustworthy, and thus are given a smaller penalty than rare ones. Paraphrase penalties are weighted by the constant parameter λ. which controls the amount of paraphrasing we allow in the output. The objective function is the sum of the salience scores and paraphrase penalties of all the phrases chosen to form the output of a given document, subject to the constraints in Equations (1b)–(1j). The latter provide a natural way of describing the requirements the output must meet.

Constraints (1b) and (1c) ensure that the generated output stays within the acceptable length range of (L_{min}, L_{max}) tokens. Equation (1d) is a set-covering constraint, requiring that at least T_{min} words in $\mathcal{T}$ appear in the output. This is important where we want to focus on some aspect of the source document, for instance on the subject of an image.

Constraint (1e) ensures that the phrase dependencies are respected and thus enforces grammatical correctness. Phrases that depend on phrase i are contained in the set $\mathcal{D}_i$. Variable x_i is true, and therefore phrase i will be included, if any of its dependents $x_j \in \mathcal{D}_i$ are true. The phrase dependency constraints, contained in the set $\mathcal{D}_i$ and enforced by (1e), are the result of three principles based on the typed dependency information:

1. Where the QG provides alternative paraphrases, it makes sense of course to select only one. This is controlled by constraint (1f), and by placing all paraphrases in the set $\mathcal{D}_i$ for the choice node i.

2. Where there are no applicable QG rules to guide the model, in general we require all child nodes j of the current node i to be included in the summary if node i is included. As exceptions, we allow the subtree represented by node j to be deleted if the dependency label for the connecting edge $i \rightarrow j$ is of type *advcl* (adverbial clause) or some form of *conj* (conjunction).

3. In general, we force the parent node p of the current node i to be included in the output if i is, resulting in all ancestors up to the root node being included. We allow a break, and the subtree at i to be used as a stand-alone sentence, if the PCFG parser has marked i with an S (sentence) label.

Constraint (1g) tells the ILP to output a sentence if one of its constituent phrases is chosen. Finally, (1h) limits the output to a maximum of N_S sentences.

4 Experimental Set-up

As mentioned earlier we evaluated the performance of our model on two title generation tasks, namely headline and caption generation. In this section we give details on the corpora and grammars we used, model parameters and features. We also describe the baselines used for comparison with our approach, and explain how system output was evaluated.

Training We obtained phrase-based salience scores using a supervised machine learning algorithm. For the headline generation task, the full DUC-03 (Task 1) corpus was used for training; it contains 500 documents and 4 headline-style summaries per document. For the captions, training data was gathered from the CNN news website.[2] We used 200 documents and their corresponding captions. Sentences were first tokenized to separate words and punctuation, and then parsed to obtain phrases and dependencies as described in Section 3 using the Stanford parser (Klein and Manning, 2003). Document phrases were marked as positive or negative automatically. If there was a unigram overlap (excluding stop words) between the phrase and any of the original title or caption, we marked this phrase with a positive label. Non-overlapping phrases were given negative labels.

Our feature set comprised surface features such as sentence and paragraph position information, POS tags, and whether high-scoring tf.idf words were present in the phrase. Additionally, the caption training set contained features for unigram and bigram overlap with the title. We learned the feature weights with a linear SVM, using the software SVM-OOPS (Woodsend and Gondzio, 2009). This tool gave us directly the feature weights as well as support vector values, and it allowed different penalties to be applied to positive and negative misclassifications, enabling us to compensate for the unbalanced data set. The penalty hyper-parameters chosen were the ones that gave the best F-scores, using 10-fold validation.

For each of the two tasks, QG rules were extracted from the same data used to train the SVM, resulting in 2,910 distinct rules for headlines and 2,757 rules for the captions. Table 1 shows that for both tasks, the majority of rules apply to PP and NP phrases. Both tasks involve considerable compression, but the proportions of the rewrite operations involved indicate differences in style between them. Compared

[2] See `http://edition.cnn.com/`.

Label	Prop'n	Proportion for Label			
	of set	Unmod	Del	Ins	Re-ord
PP	40%	5%	93%	12%	6%
NP	31%	5%	87%	14%	7%
S	20%	1%	96%	15%	7%
SBAR	6%	4%	95%	28%	6%

(a) Headlines

Label	Prop'n	Proportion for Label			
	of set	Unmod	Del	Ins	Re-ord
PP	30%	17%	81%	7%	4%
NP	29%	17%	76%	11%	3%
S	27%	10%	84%	16%	6%
SBAR	10%	13%	80%	16%	3%

(b) Captions

Table 1: QG rules generated for (a) headline and (b) caption tasks (top 4 labels shown). The columns show label of root node, proportion of the full ruleset, then the proportions of rules for this label involving no modification, deletions, insertions and re-orderings.

to headlines, captions involve slightly less deletion and a higher proportion of the phrases are unmodified. The QG learning mechanism also discovers more alignments between source sentences and captions than it does for the headline task.

Title generation For the headline generation task, we evaluated our model on a testing partition from the DUC-04 corpus (75 documents, Task 1). For the caption task, we used the test set (240 documents) described in Feng and Lapata (2010a). Their corpus was downloaded from the BBC news site and contains documents, images, and their captions.[3]

We created and solved an ILP for each document. For each phrase, features were extracted and salience scores calculated from the feature weights determined through SVM training. The distance from the SVM hyperplane represents the salience score. Parameters for the ILP models for the two tasks are shown in Table 2. The λ parameter was set to 0.2 to ensure that paraphrases were included; other parameters were chosen to capture the properties seen in the majority of the training set. Note the maximum number of sentences allowed to form a headline is set to 5 as some of the headlines in the DUC dataset contained multiple sentences.

Parameter		Headlines	Captions
Min length	L_{min}	8	8
Max length	L_{max}	16	20
Min keywords	T_{min}	0	2
Max sentences	N_S	5	1
Paraphrase	λ	0.2	0.1

Table 2: ILP model parameters for the two tasks.

To solve the ILP model we used the ZIB Optimization Suite software (Achterberg, 2007; Koch, 2004). The solution was converted into a sentence by removing nodes not chosen from the tree representation, then concatenating the remaining leaf nodes in order.

Model Comparison For the headline task, we compared our model to the DUC-04 standard baseline of the first sentence, truncated at the first word boundary after 75 characters; and the output of the Topiary system (Zajic et al., 2004), which came top in almost all measures in the DUC-04 evaluation. In order to generate a headline, Topiary first compresses the lead sentence using linguistically motivated heuristics and then enhances it with topic keywords. For the captions, we compared our model against the highest-scoring document sentence according to the SVM and against the probabilistic model presented in Feng and Lapata (2010a). The latter estimates the probability of a phrase appearing in the caption given the same phrase appearing in the corresponding document and uses a language model to select among many different surface realizations. The language model is adapted with probabilities from an image annotation model (Feng and Lapata, 2010b).

Evaluation We evaluated the quality of the headlines using ROUGE (Lin and Hovy, 2003). The DUC-04 dataset provides four reference headlines per document. We report unigram overlap (ROUGE-1) and bigram overlap (ROUGE-2) as a means of assessing informativeness, and the longest common subsequence (ROUGE-L) as a means of as-

[3] Available from `http://homepages.inf.ed.ac.uk/s0677528/data.html`.

sessing fluency. Original DUC-04 ROUGE parameters were used. We also use ROUGE to evaluate the automatic captions with the original BBC captions as reference.

In addition, we evaluated the generated headlines by eliciting human judgments. Participants were presented with a news article and its corresponding headline and were asked to rate the latter along two dimensions: informativeness (does the headline capture the article's most important information?), and grammaticality (is it fluent and easy to understand?). The subjects used a seven point rating scale; an ideal system would receive high numbers for both measures. We randomly selected twelve documents from the test set and generated headlines with our model. We also included the output of Topiary and the human written DUC-04 headlines as a gold standard. We thus obtained ratings for 48 (12 × 4) document-highlights pairs.

We elicited judgments for the generated captions in a similar fashion. Participants were presented with a document, an associated image, and its caption, and asked to rate the latter (using a 1–7 rating scale) with respect to grammaticality and informativeness (does it describe succinctly the content of the image and document?). Again, we randomly selected 12 document-image pairs from the test set and generated captions for them using the highest scoring document sentence according to the SVM, our ILP-based model, and the output of Feng and Lapata's (2010a) system. We also included the original BBC captions as an upper bound. Both studies were conducted over the Internet using WebExp (Keller et al., 2009). 80 unpaid volunteers rated the headlines and 65 the captions, all self reported native English speakers.

5 Results

We report results on the headline generation task in Figure 3, with ROUGE-1, ROUGE-2 and ROUGE-L. In ROUGE-1 and ROUGE-L measures, the best scores are obtained by the Topiary system, slightly better than the lead sentence baseline, while for ROUGE-2 the ordering is reversed. Our model does not outperform the lead sentence or Topiary. Note that the 95% confidence level intervals reported by ROUGE are so large that no results are statistically

Lead	The chances for a new, strictly secular government in Turkey faded Wednesday.
Topiary	TURKEY YILMAZ PARTY ECEVIT chances strictly secular government faded.
ILP	Bulent Ecevit needs Turkey's two-center right parties to hammer together secular coalition.
DUC	Chance for new, secular, Turkish government fades; what will Ecevit do now?
Source	Premier-designate Bulent Ecevit needs Turkey's two-center right parties to hammer together a secular coalition, but Tansu Ciller, the ex-premier who commands 99 votes in parliament, rebuffed him Wednesday.
Lead	U.S. President Bill Clinton won South Korea's support Saturday for confronting.
Topiary	NUCLEAR U.S. President Bill Clinton won for confronting North Korea.
ILP	North Koreans have denied construction site has nuclear purpose.
DUC	U.S. warns N. Korea not to waste chance for peace over alleged nuclear site.
Source	The North Koreans have denied the underground construction site has any nuclear purpose, and it has demanded a dlrs 300 million payment for proving that.
Lead	By only one vote, the center-left prime minister of Italy, Romano Prodi.
Topiary	PRODI By only one vote center left prime minister and toppled from power.
ILP	Political system changes, Italy is condemned to political instability.
DUC	Prodi loses confidence vote; will stay as caretaker until new government.
Source	"Unless the Italian political system changes, Italy is condemned to political instability," said Sergio Romano, a former diplomat and political science professor.

Table 3: Example headline output.

F&L	The former paramedic training officer stood at the next general election.
ILP	The majority are now believing that war in Iraq was wrong.
BBC	L/Cpl Thomas Keys was shot 18 times, his inquest heard.
Source	The majority of people in this country are now believing that the war in Iraq was wrong, and I do believe we will get support.
F&L	The state government of Victoria take as those tests for cannabis.
ILP	Police in Victoria have begun randomly testing drivers for the drug ecstasy.
BBC	Police say drugs like Ecstasy can be as dangerous as alcohol for drivers.
Source	Police in the Australian state of Victoria have begun randomly testing drivers for the drug ecstasy.
F&L	The US Government Professor Holdren called for more than a year.
ILP	"We are experiencing dangerous human disruption of global climate," Professor Holdren said.
BBC	Sea levels could rise by 4m over the coming century, he warns.
Source	"We are experiencing dangerous human disruption of the global climate and we're going to experience more," Professor Holdren said.

Table 4: Example caption output.

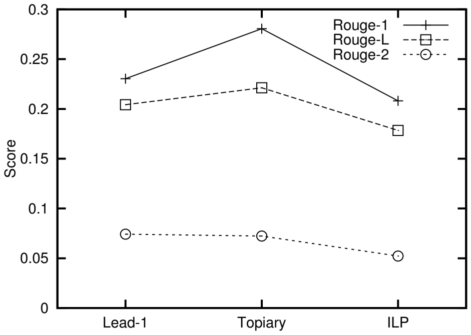


Figure 3: ROUGE-1, ROUGE-2 and ROUGE-L results on the DUC-04 headlines for our ILP model, the lead sentence baseline and Topiary.

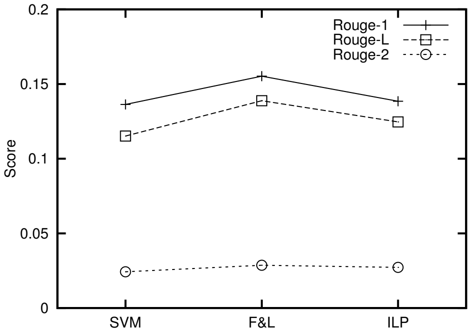


Figure 4: ROUGE-1, ROUGE-2 and ROUGE-L results on the BBC captions for our ILP model, the sentence baseline chosen by the SVM, and Feng and Lapata's (2010) model.

Model	Grammaticality	Importance
Lead-1	4.95	3.30
Topiary	3.03	3.43
ILP	5.36	4.94
Reference	5.12	5.17

Table 5: Average human ratings of DUC-04 headlines, for our ILP model, the lead sentence baseline, the output of Topiary and the human-written reference.

Model	Grammaticality	Importance
SVM	5.24	5.01
F&L	4.42	4.74
ILP	5.49	5.25
Reference	5.61	5.18

Table 6: Average human ratings of captions, for our ILP model, the sentence baseline chosen by the SVM, Feng and Lapata's (2010) model and the reference BBC caption.

significant. We also investigated using an ILP model with just the QG rules or just dependency label information (see constraint (1e) in Section 3.3). Both settings gave less compressed output, and the resulting ROUGE scores were lower on all measures. The ROUGE results for the caption generation task follow a similar pattern (see Figure 4). Our model is slightly better than the best sentence baseline but performs worse than Feng and Lapata (2010a). Tables 3 and 4 show example output for the ILP model and the baselines on the headline and caption tasks respectively. In the tables, *Source* refers to the sentence chosen by the ILP, but before any paraphrasing is applied. We can see that deletion rules dominate, and a more compressive style of paraphrasing has been learned for the headline task.

The results of our human evaluation study for the DUC-04 headlines are summarized in Table 5. Means differences were compared using a Post-hoc Tukey test. The headlines created by our model were considered significantly more important and more grammatical than those of the Topiary system ($\alpha < 0.01$), despite the better overlap of Topiary with the reference headlines as indicated in the Rouge results above. Compared to the lead sentence of the article (the DUC-04 baseline), our model was also rated significantly higher in terms of importance ($\alpha < 0.01$) but not grammaticality.

Table 6 summarizes the results of our second judgment elicitation study. The captions generated by our model are significantly more grammatical than those of Feng and Lapata (2010a) ($\alpha < 0.01$). The SVM, ILP model and reference captions do not differ significantly in terms of grammaticality. In terms of importance, the ILP model is significantly better than the SVM ($\alpha < 0.01$) and Feng and Lapata ($\alpha < 0.01$) and comparable to the reference.

The human ratings are more favorable to our model than ROUGE for both tasks. There are two reasons for this. Firstly, the model is not biased towards selecting the lead sentence as a headline/caption and is disadvantaged in ROUGE evaluations as professional abstractors often reuse the lead or parts of it to create a title. Secondly, the model often generates an appropriate title that is lexically distinct from the reference even though it expresses similar meaning.

6 Conclusions

In this paper we proposed a joint content selection and surface realization model for single-document summarization. The model operates over a syntax-rich representation of the source document and learns which phrases should be in the summary. Content selection preferences are coupled with a quasi-synchronous grammar whose rules encode surface realization preferences (e.g., paraphrases and compressions). Both types of preferences are optimized simultaneously in an integer linear program subject to grammaticality, length and coverage constraints. Importantly, the QG allows the model to adapt to the writing and stylistic conventions of different tasks. The results of our human studies show that our system creates grammatical and informative summaries whilst outperforming several competitive baselines.

The model itself is relatively simple and achieves good performance without any task-specific modification. One potential stumbling block may be the availability of parallel data for acquiring the QG. The Internet provides a large repository of news documents with headlines, images and captions. In some cases news articles are even accompanied with "story highlights" which could be used as training data for longer summaries.[4] For other domains obtaining such data may be more difficult. However, our experiments have shown that relatively small parallel corpora (in the range of 200–500 pairs) suffice to learn many of the writing conventions for a given task.

In the future, we plan to explore how to integrate more sophisticated QG rules in the generation process. Currently we consider deletions, reorderings and insertions. Ideally, we would also like to model arbitrary substitutions between words but also larger constituents (e.g., subclauses, sentence aggregation). Beyond summarization, we would also like to apply our model to other generation tasks, such as paraphrasing and text simplification.

Acknowledgments We are grateful to David Chiang and Noah Smith for their input on earlier versions of this work. We would also like to thank Andreas Grothey and members of ICCS at the School of Informatics for valuable discussions and comments. We acknowledge the support of EPSRC through project grants EP/F055765/1 and GR/T04540/01.

References

Achterberg, Tobias. 2007. *Constraint Integer Programming*. Ph.D. thesis, Technische Universität Berlin.

Banko, Michele, Vibhu O. Mittal, and Michael J. Witbrock. 2000. Headline generation based on statistical translation. In *Proceedings of the 38th ACL*. Hong Kong, pages 318–325.

Clarke, James and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research* 31:399–429.

Cohn, Trevor and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of the 22nd COLING*. Manchester, UK, pages 137–144.

Das, Dipanjan and Noah A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the ACL-IJCNLP*. Suntec, Singapore, pages 468–476.

Daumé III, Hal. 2006. *Practical Structured Learning Techniques for Natural Language Processing*. Ph.D. thesis, University of Southern California.

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

Dorr, Bonnie, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 2003 Text Summarization Workshop and Document Understanding Conference*. Edmondon, Alberta, pages 1–8.

Dras, Mark. 1999. *Tree Adjoining Grammar and the Reluctant Paraphrasing of Text.*. Ph.D. thesis, Macquarie University.

Feng, Yansong and Mirella Lapata. 2010a. How many words is a picture worth? Automatic caption generation for news images. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Uppsala, Sweden, pages 1239–1249.

Feng, Yansong and Mirella Lapata. 2010b. Topic models for image annotation and text illustration. In *Pro-*

[4] On-line CNN news articles are prefaced by story highlights—three or four short sentences that are written by humans and give a brief overview of the article.

ceedings of the NAACL HLT. Association for Computational Linguistics, Los Angeles, California, pages 831–839.

Jing, Hongyan. 2000. Sentence reduction for automatic text summarization. In *Proceedings of the 6th ANLP*. Seattle, WA, pages 310–315.

Jing, Hongyan. 2002. Using hidden Markov modeling to decompose human-written summaries. *Computational Linguistics* 28(4):527–544.

Jing, Hongyan and Kathleen McKeown. 2000. Cut and paste summarization. In *Proceedings of the 1st NAACL*. Seattle, WA, pages 178–185.

Keller, Frank, Subahshini Gunasekharan, Neil Mayo, and Martin Corley. 2009. Timing accuracy of web experiments: A case study using the WebExp software package. *Behavior Research Methods* 41(1):1–12.

Klein, Dan and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st ACL*. Sapporo, Japan, pages 423–430.

Koch, Thorsten. 2004. *Rapid Mathematical Prototyping*. Ph.D. thesis, Technische Universität Berlin.

Kupiec, Julian, Jan O. Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of SIGIR-95*. Seattle, WA, pages 68–73.

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

Lin, Chin-Yew and Eduard H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of HLT NAACL*. Edmonton, Canada, pages 71–78.

Martins, André and Noah A. Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing*. Boulder, Colorado, pages 1–9.

Smith, David and Jason Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings on the Workshop on Statistical Machine Translation*. Association for Computational Linguistics, New York City, pages 23–30.

Smith, David A. and Jason Eisner. 2009. Parser adaptation and projection with quasi-synchronous grammar features. In *Proceedings of the EMNLP*. Suntec, Singapore, pages 822–831.

Soricut, R. and D. Marcu. 2007. Abstractive headline generation using WIDL-expressions. *Information Processing and Management* 43(6):1536–1548. Text Summarization.

Wang, Mengqiu, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proceedings of the EMNLP-CoNLL*. Prague, Czech Republic, pages 22–32.

Woodsend, Kristian and Jacek Gondzio. 2009. Exploiting separability in large-scale linear support vector machine training. *Computational Optimization and Applications* Published online.

Woodsend, Kristian and Mirella Lapata. 2010. Automatic generation of story highlights. In Sandra Carberry and Stephen Clark, editors, *Proceedings of the 48th ACL*. Uppsala, Sweden, pages 565–574.

Zajic, David, Bonnie Dorr, and Richard Schwartz. 2004. BBN/UMD at DUC-2004: Topiary. In *Proceedings of the NAACL Workshop on Document Understanding*. Boston, MA, pages 112–119.

Zajic, David, Bonnie J. Dorr, Jimmy Lin, and Richard Schwartz. 2007. Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. *Information Processing Management Special Issue on Summarization* 43(6):1549–1570.

Zhao, Shiqi, Xiang Lan, Ting Liu, and Sheng Li. 2009. Application-driven statistical paraphrase generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore, pages 834–842.

Automatic Analysis of Rhythmic Poetry with Applications to Generation and Translation

Erica Greene
Haverford College
370 Lancaster Ave.
Haverford, PA 19041
ericagreene@gmail.com

Tugba Bodrumlu
Dept. of Computer Science
Univ. of Southern California
Los Angeles, CA 90089
bodrumlu@cs.usc.edu

Kevin Knight
Information Sciences Institute
Univ. of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292
knight@isi.edu

Abstract

We employ statistical methods to analyze, generate, and translate rhythmic poetry. We first apply unsupervised learning to reveal word-stress patterns in a corpus of raw poetry. We then use these word-stress patterns, in addition to rhyme and discourse models, to generate English love poetry. Finally, we translate Italian poetry into English, choosing target realizations that conform to desired rhythmic patterns.

1 Introduction

When it comes to generating creative language (poems, stories, jokes, etc), people have massive advantages over machines:

- people can construct grammatical, sensible utterances,
- people have a wide range of topics to talk about, and
- people experience joy and heart-break.

On the other hand, machines have some minor advantages:

- a machine can easily come up with a five-syllable word that starts with *p* and rhymes with *early*, and
- a machine can analyze very large online text repositories of human works and maintain these in memory.

In this paper we concentrate on statistical methods applied to the analysis, generation, and translation of poetry. By analysis, we mean extracting patterns from existing online poetry corpora. We use these patterns to generate new poems and translate existing poems. When translating, we render target text in a rhythmic scheme determined by the user.

Poetry generation has received research attention in the past (Manurung et al., 2000; Gervas, 2001; Diaz-Agudo et al., 2002; Manurung, 2003; Wong and Chun, 2008; Tosa et al., 2008; Jiang and Zhou, 2008; Netzer et al., 2009), including the use of statistical methods, although there is a long way to go. One difficulty has been the evaluation of machine-generated poetry—this continues to be a difficulty in the present paper. Less research effort has been spent on poetry analysis and poetry translation, which we tackle here.

2 Terms

Meter refers to the rhythmic beat of poetic text when read aloud. Iambic is a common meter that sounds like *da-DUM da-DUM da-DUM*, etc. Each *da-DUM* is called a foot. Anapest meter sounds like *da-da-DUM da-da-DUM da-da-DUM*, etc.

Trimeter refers to a line with three feet, pentameter to a line with five feet, etc. Examples include:

- a VE-ry NAS-ty CUT (iambic trimeter)
- shall I com-PARE thee TO a SUM-mer's DAY? (iambic pentameter)
- twas the NIGHT before CHRIST-mas and ALL through the HOUSE (anapest tetrameter)

Classical English sonnets are poems most often composed of 14 lines of iambic pentameter.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 524–533,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

3 Analysis

We focus on English rhythmic poetry. We define the following analysis task: *given poetic lines in a known meter (such as sonnets written in iambic pentameter), assign a syllable-stress pattern to each word in each line*. Making such decisions is part of the larger task of reading poetry aloud. Later in the paper, we will employ the concrete statistical tables from analysis to the problems of poetry generation and translation.

We create a test set consisting of 70 lines from Shakespeare's sonnets, which are written in iambic pentameter. Here is an input line annotated with gold output.

```
shall i compare thee to a summers day
|     |    /\    |    |  |   /\     |
S     S*  S  S* S    S* S  S* S    S*
```

S refers to an unstressed syllable, and S* refers to a stressed syllable. One of the authors created gold-standard output by listening to Internet recordings of the 70 lines and marking words according to the speaker's stress. The task evaluation consists of *per-word accuracy* (how many words are assigned the correct stress pattern) and *per-line accuracy* (how many lines have all words analyzed perfectly).

This would seem simple enough, if we are armed with something like the CMU pronunciation dictionary: we look up syllable-stress patterns for each word token and lay these down on top of the sequence S S* S S* S S* S S* S S*. However, there are difficulties:

- The test data contains many words that are unknown to the CMU dictionary.
- Even when all words are known, many lines do not seem to contain 10 syllables. Some lines contain eleven words.
- Spoken recordings include stress reversals, such as poin-TING instead of POIN-ting.
- Archaic pronunciations abound, such as PROV-ed (two syllables) instead of PROVED (one syllable).
- In usage, syllables are often subtracted (PRIS-ner instead of PRIS-o-ner), added (SOV-e-reign instead of SOV-reign), or merged.
- Some one-syllable words are mostly stressed, and others mostly unstressed, but the dictionary provides no guidance. When we generate rhythmic text, it is important to use one-syllable words properly. For example, we would be happy for an iambic generator to output *big thoughts are not quite here*, but not *quite big thoughts are not here*.

$$\text{e} \rightarrow \boxed{\text{P}(m|e)} \rightarrow \text{m}$$

Figure 1: Finite-state transducer (FST) for mapping sequences of English words (e) onto sequences of S* and S symbols (m), representing stressed and unstressed syllables.

Therefore, we take a different tack and apply unsupervised learning to acquire word-stress patterns directly from raw poetry, without relying on a dictionary. This method easily ports to other languages, where dictionaries may not exist and where morphology is a severe complication. It may also be used for dead languages.

For raw data, we start with all Shakespeare sonnets (17,134 word tokens). Because our learning is unsupervised, we do not mind including our 70-line test set in this data (*open testing*).

Figures 1 and 2 show a finite-state transducer (FST) that converts sequences of English words to sequences of S* and S symbols. The FST's transitions initially map each English word onto all output sub-sequences of lengths 1 to 4 (i.e., S, S*, S-S, S-S*, S*-S, S*-S*, S-S-S, . . .) plus the sequences S-S*-S-S*-S and S*-S-S*-S-S*. Initial probabilities are set to 1/32. The FST's main loop allows it to process a sequence of word tokens. If the same word appears twice in a sequence, then it may receive two different pronunciations, since the mapping is probabilistic. However, a token's syllable/stress pattern is chosen independently of other tokens in the sequence; we look at relaxing this assumption later.

We next use finite-state EM training[1] to train the machine on input/output sequences such as these:

```
from fairest creatures we desire increase
S S* S S* S S* S S* S S*

but thou contracted to thine own bright eyes
S S* S S* S S* S S* S S*
```

[1] All operations in this paper are carried out with the generic finite-state toolkit Carmel (Graehl, 1997). For example, the *train-cascade* command uses EM to learn probabilities in an arbitrary FST cascade from end-to-end input/output string pairs.

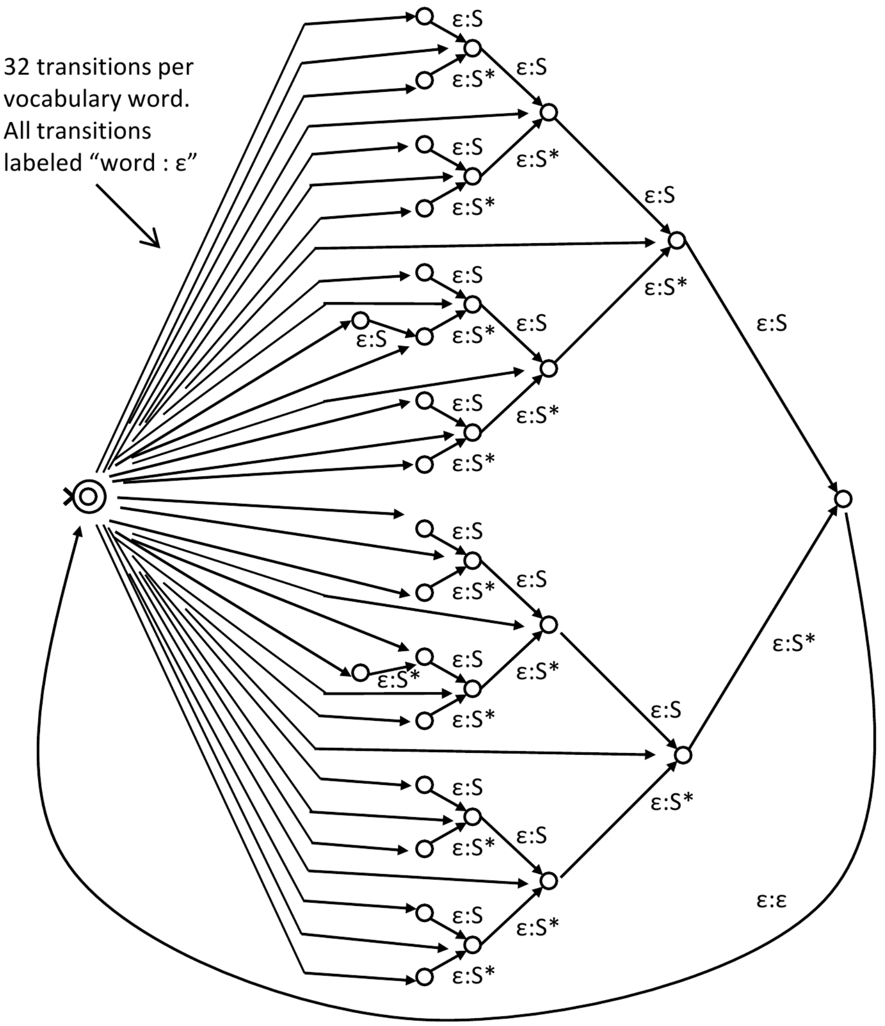


Figure 2: An efficient FST implementing P($m|e$). This machine maps sequences of English words onto sequences of S* and S symbols, representing stressed and unstressed syllables. Initially every vocabulary word has 32 transitions, each with probability 1/32. After EM training, far fewer transitions remain.

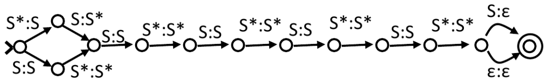


Figure 3: An FST that accepts any of four input meters and deterministically normalizes its input to strict iambic pentameter. We call this FST *norm*.

$$e \rightarrow \boxed{\mathrm{P}(m|e)} \rightarrow \mathrm{m} \rightarrow \boxed{norm} \rightarrow \mathrm{m}$$

Figure 4: FST cascade that encodes a loose interpretation of iambic pentameter. The *norm* FST accepts any of four near-iambic-pentameter sequences and normalizes them into strict iambic pentameter.

Note that the output sequences are all the same, representing our belief that each line should be read as iambic pentameter.[2] After we train the FST, we can use Viterbi decoding to recover the highest-probability alignments, e.g.:

```
from fairest creatures we desire increase
 |     |       /| \    |    /\      /\
 S     S*     S S* S   S*  S  S*   S  S*

but thou contracted to thine own bright eyes
 |    |    /| \     |    |    |    |     |
 S    S*  S S* S    S*   S    S*   S     S*
```

Note that the first example contains an error—the words *fairest* and *creatures* should each be read with two syllables. There are many such errors. We next improve the system in two ways: more data and better modeling.

First, we augment the Shakespeare sonnets with data from the website *sonnets.org*, increasing the number of word tokens from 17,134 to 235,463. The *sonnets.org* data is noisier, because it contains some non-iambic-pentameter poetry, but overall we find that alignments improve, e.g.:

```
from fairest creatures we desire increase
 |     /\       /\     |    /\      /\
 S    S* S     S* S    S*  S  S*   S  S*
```

Second, we loosen our model. When we listen to recordings, we discover that not all lines are read S S* S S* S S* S S* S S*. Indeed, some lines in our data contain eleven words—these are unexplainable by the EM training system. We also observe that poets often use the word *mother* (S* S) at the beginnings and ends of lines, where it theoretically should not appear.

Training data	Training tokens	Test token accuracy	Test line accuracy
Shakespeare	17,134	82.3%	55.7%
sonnets.org	235,463	94.2%	81.4%

Figure 5: Analysis task accuracy.

Two well-known variations explain these facts. One is optional *inversion* of the first foot (S S* → S* S). Second is the optional addition of an eleventh unstressed syllable (the *feminine ending*). These variations yield four possible syllable-stress sequences:

```
S  S* S S* S S* S S* S S*
S* S  S S* S S* S S* S S*
S  S* S S* S S* S S* S S* S
S* S  S S* S S* S S* S S* S
```

We want to offer EM the freedom to analyze lines into any of these four variations. We therefore construct a second FST (Figure 3), $norm$, which maps all four sequences onto the canonical pattern S S* S S* S S* S S* S S*. We then arrange both FSTs in a cascade (Figure 4), and we train the whole cascade on the same input/output sequences as before. Because $norm$ has no trainable parameters, we wind up training only the lexical mapping parameters. Viterbi decoding through the two-step cascade now reveals EM's proposed internal meter analysis as well as token mappings, e.g.:

```
to be or not to be that is the question
|  |  |   |  |  |    |  |   |     /\
S  S* S   S* S  S*   S  S*  S    S* S
|  |  |   |  |  |    |  |   |    |
S  S* S   S* S  S*   S  S*  S    S*
```

Figure 5 shows accuracy results on the 70-line test corpus mentioned at the beginning of this section. Over 94% of word tokens are assigned a syllable-stress pattern that matches the pattern transcribed from audio. Over 81% of whole lines are also scanned correctly. The upper limit for whole-line scanning under our constraints is 88.6%, because 11.4% of gold outputs do not match any of the four patterns we allow.

We further obtain a probabilistic table of word mappings that we can use for generation and trans-

[2] We can augment the data with lines of poetry written in meters other than iambic pentameter, so long as we supply the desired output pattern for each input line.

```
P(S* S S* | altitude) = 1.00

P(S* S | creatures)   = 1.00

P(S* S  | pointed)    = 0.95
P(S  S* | pointed)    = 0.05

P(S* S    | prisoner) = 0.74
P(S* S S* | prisoner) = 0.26

P(S* S  | mother)     = 0.95
P(S*    | mother)     = 0.03
P(S  S* | mother)     = 0.02
```

Figure 6: Sample learned mappings between words and syllable-stress patterns.

word	P(S* | word)	P(S | word)
a	0.04	0.96
the	0.06	0.94
their	0.09	0.91
mens	0.10	0.90
thy	0.10	0.90
be	0.48	0.52
me	0.49	0.51
quick	0.50	0.50
split	0.50	0.50
just	0.51	0.49
food	0.90	0.10
near	0.90	0.10
raised	0.91	0.09
dog	0.93	0.07
thought	0.95	0.05

Figure 7: Sample mappings for one-syllable words.

lation tasks. Figure 6 shows a portion of this table. Note that P(S S* | mother) has a very small probability of 0.02. We would incorrectly learn a much higher value if we did not loosen the iambic pentameter model, as many *mother* tokens occur line-initial and line-final.

Figure 7 shows which one-syllable words are more often stressed (or unstressed) in iambic pentameter poetry. Function words and possessives tend to be unstressed, while content words tend to be stressed, though many words are used both ways. This useful information is not available in typical pronunciation dictionaries.

$$\boxed{\mathrm{P}(m)} \rightarrow \mathrm{m} \rightarrow \boxed{\mathrm{P}(e|m)} \rightarrow \mathrm{e} \rightarrow \boxed{\mathrm{P}(e)} \rightarrow \mathrm{e}$$

Figure 8: Finite-state cascade for poetry generation.

Alignment errors still occur, especially in noisy portions of the data that are not actually written in iambic pentameter, but also in clean portions, e.g.:

```
the perfect ceremony of loves rite
 |     /\      /|\    |   |    /\
 S    S* S    S* S S*  S   S*  S  S*
```

The word *ceremony* only occurs this once in the data, so it is willing to accept any stress pattern. While *rite* is correctly analyzed elsewhere as a one-syllable word, *loves* prefers S*, and this overwhelms the one-syllable preference for *rite*. We can blame our tokenizer for this, as it conflates *loves* and *love's*, despite the fact that these words have different stress probabilities.

4 Generation

Figure 8 shows our concept of generation as a cascade of weighted FSTs.

P(m) is a user-supplied model of desired meters—normally it deterministically generates a single string of S* and S symbols. (The user also supplies a rhyme scheme—see below).

P($e|m$) is the reverse of Section 3's P($m|e$), being a model of word selection. Its generative story is: (1) probabilistically select n tokens (n = 1 to 5) from the input, (2) probabilistically select a word w that realizes that n-token sequence, and (3) recurse until the input is consumed. Instead of asking how a given word is likely to be pronounced (e.g., S or S*), we now ask how a given stress-pattern (e.g., S or S*) is likely to be realized. This model is trained with the same method described in Section 3 and is augmented with the CMU pronunciation dictionary.

Finally, P(e) is a word-trigram model built from a 10,000-line corpus of 105 English love poems.

We select the first line of our poem from the FST cascade's 100,000-best list, or by hand. To generate each subsequent line, we modify the cascade and run it again. The first modification is to incorporate a discourse model. From our poetry corpus, we estimate a word's unigram probability given the words on the previous line, via IBM Model 1 (Brown et al., 1993). We modify P(e) by interpolating in these probabilities. Second, we check if any previous line

```
The women of the night
Again and all the way
Like a mouse in the white
Not the heart of the day.
  - - -
Of the bed to trust me
Around her twists the string
But i will not tell thee
Fire changes everything.
  - - -
A son of the right hand confines
His uncle could have broken in
Towards the high bank and the pines
Upon the eyes and i have been
  - - -
Into one of her hundred year old
Or the house in a house in a cold
The first time she met him
Like a mouse in the dim
For me to the moon and when i told
  - - -
Into one of them some years before
His own man or the house in a more
The moon and when the day
Into one of the way
With the breath from the first time
  she swore
```

Figure 9: Sample poems generated with a weighted FST cascade.

$w_1, w_2, ...w_n$ needs to be rhymed with, according to the user-supplied scheme. If so, we build an additional FST that accepts only strings whose final word rhymes with w_n. This is a reasonable approach, though it will not, for example, rhyme *...tar me* with *...army*. We say two non-identical words rhyme if their phoneme strings share a common suffix that includes the last stressed vowel.

Figure 9 shows several poems that we automatically generate with this scheme.

5 Translation

Automatically generated poetry can sound good when read aloud, but it often has a "nonsense" feel to it. According to (Gervas, 2010), creative-language researchers interested in realization and surface language statistics ("how to say") have tended to gravitate to poetry generation, while researchers interested in characters, goals, and story-line ("what to say") have tended to gravitate to prose story generation.

Translation provides one way to tie things together. The source language provides the input ("what to say"), and the target language can be shaped to desired specifications ("how to say"). For example, we may want to translate Italian sonnets into fluent English iambic pentameter. This is certainly a difficult task for people, and one which is generally assumed to be impossible for computers.

$$\mathrm{i} \rightarrow \boxed{\mathrm{P}(e|i)} \rightarrow \mathrm{e} \rightarrow \boxed{\mathrm{P}(m|e)} \rightarrow \mathrm{m} \rightarrow \boxed{\mathrm{P}(m)} \rightarrow \mathrm{m}$$

Figure 10: Finite-state cascade for poetry translation.

Here we investigate translating Dante's *Divine Comedy* (DC) from Italian into English by machine. The poem begins:

```
nel mezzo del cammin di nostra vita
mi ritrovai per una selva oscura
che la via diritta era smarrita.
```

DC is a long sequence of such three-line stanzas (*tercets*). The meter in Italian is hendecasyllabic, which has ten syllables and ensures three beats. Dante's Italian rhyme scheme is: **ABA**, **BCB**, **CDC**, etc, meaning that lines 2, 4, and 6 rhyme with each other; lines 5, 7, and 9 rhyme with each other, and so forth. There is also internal rhyme (e.g., *diritta/smarrita*).

Because DC has been translated many times into English, we have examples of good outputs. Some translations target iambic pentameter, but even the most respected translations give up on rhyme, since English is much harder to rhyme than Italian. Longfellow's translation begins:

```
midway upon the journey of our life
i found myself within a forest dark
for the straightforward pathway had
  been lost.
```

We arrange the translation problem as a cascade of WFSTs, as shown in Figure 10. We call our Italian input i. In lieu of the first WFST, we use the statistical phrase-based machine translation (PBMT) system Moses (Koehn et al., 2007), which generates a target-language lattice with paths scored by P($e|i$). We send this lattice through the same P($m|e$) device we trained in Section 3. Finally, we filter the resulting syllable sequences with a strict, single-path, deterministic iambic pentameter acceptor, P(m).[3] Our

[3] It is also possible to use a looser iambic P(m) model, as described in Section 3.

Parallel Italian/English Data

Collection	Word count (English)
DC-train	400,670
Il Fiore	25,995
Detto Damare	2,483
Egloghe	3,120
Misc.	557
Europarl	*32,780,960*

English Language Model Data

Collection	Word count (English)
DC-train	400,670
poemhunter.com poetry.eserver.org poetrymountain.com	686,714
poetryarchive.org	58,739
everypoet.com	574,322
sonnets.org	166,465
Europarl	*32,780,960*

Tune and Blind Test Data (4 reference)

Collection	Word count (Italian)
DC-tune	7,674
DC-test	2,861

Figure 11: Data for Italian/English statistical translation.

finite-state toolkit's top-k paths represent the translations with the highest product of scores P($e|i$) · P($m|e$) · P(m).

In general, the P($e|i$) and P($m|e$) models fight each other in ranking candidate outputs. In experiments, we find that the P($e|i$) preference is sometimes so strong that the P($m|e$) model is pushed into using a low-probability word-to-stress mapping. This creates output lines that do not scan easily. We solve this problem by assigning a higher weight to the P($m|e$) model.[4]

Figure 11 shows the data we used to train the PBMT system. The vast majority of parallel Italian/English poetry is DC itself, for which we have four English translations. We break DC up into DC-train, DC-tune, and DC-test. We augment our target language model with English poetry collected from many sources. We also add Europarl data, which is out of domain, but which reduces the unknown word-token rate in DC-test from 9% to 6%, and the unknown word-type rate from 22% to 13%.

Original: nel mezzo del cammin di nostra vita mi ritrovai per una selva oscura che la via diritta era smarrita.
Phrase-based translation (PBMT): midway in the journey of our life i found myself within a forest dark for the straight way was lost.
PBMT + meter model: midway upon the journey of our life i found myself within a forest dark for the straightforward pathway had been lost.

Figure 12: Automatic translation of lines from Dante's *Divine Comedy*. In this test-on-train scenario, the machine reproduces lines from human translations it has seen.

We first experiment in a test-on-train scenario, where we translate parts of DC that are in our training set. This is a normal scenario in human poetry translation, where people have access to previous translations.

Figure 12 shows how we translate the first lines of DC, first using only PBMT, then using the full system. When we use the full system, we not only get an output string, but also the system's intended scan, e.g.:

```
midway upon the journey of our life
 /\     /\   |    /\    |   |   |
S  S*  S  S* S   S* S   S*  S   S*
```

The machine's translation here is the same as Longfellow's, which is in the training data. In other cases, we observe the machine combining existing translations, e.g.:

```
i:  bedi la bestia per cu io mi volsi
I5: behold the beast that made me turn aside

H1: BEHOLD THE BEAST for which i have turned back
H2: you see the beast THAT MADE ME TURN ASIDE
H3: see the beast that forced me to turn back
H4: look at the beast that drove me to turn back
```

I5 refs to the machine's iambic pentameter transla-

[4] We set this weight manually to 3.0, i.e., we raise all probabilities in the P($m|e$) model to the power of 3.0. Setting the weight too high results in lines that scan very well, but whose translation quality is low.

tion, while H1-4 refer to human translations. The machine also creates new translations:

```
i:  diro' de laltre cose chi vho scorte
I5: i shall explain the other things i saw

H1: speak will i of the other things i saw there
H2: ill also tell THE OTHER THINGS I SAW
H3: i will recount the other things i saw
H4: i here will tell the other things i saw
```

We can further change the target meter to anything we desire. To obtain iambic tetrameter (4-beat) translations, we delete the last two transitions of the P(m) model. We then get:

```
I4: in our life the journey way
    i found myself deep on dark wood
    that lost straightforward pathway had.

    ah how to say the what is hard
    this forest savage rough and stern
    the very thought renews the fear.
```

Translations and scans are uneven, but we have significant flexibility. We can even request translations that avoid the English letter A, by adding a filter to the end of the FST cascade, obtaining:

```
I5: in midst upon the journey of our life
    i found myself within the wood obscure
    <fail>
```

To steer clear of the adjective *dark* in the second line, the system switches from *forest* to *wood*, so obtain a proper scan. The third line fails because all paths through the translation lattice contain an A somewhere.

Translating blind-test data proves to be more difficult. We hold out Canto XXXI of DC's Paradiso section for testing. Figure 13 shows a portion of the translation results. The MT system handles unknown Italian words by passing them through to the output. The P($m|e$) meter model cannot process those words, accounting for the I5 failure rate.

Here, we get a first look at statistical MT translating poetry into rhythmic structures—as with all MT, there are successes and problems, and certainly more to do.

Original:

in forma dunque di candida rosa
mi si mostrava la milizia santa
che nel suo sangue cristo fece sposa

ma laltra che volando vede e canta
la gloria di colui che la nnamora
e la bonta‘ che la fece cotanta

Human translation:

in fashion then as of a snow white rose
displayed itself to me the saintly host
whom christ in his own blood had made his bride

but the other host that flying sees and sings
the glory of him who doth enamour it
and the goodness that created it so noble

Phrase-based translation (PBMT):

in the form so rose candida
i now was shown the militia holy
that in his blood christ did bride

but the other that flying sees and sings
the glory of him that the nnamora
and the goodness that the made cotanta

PBMT + meter model:

<fail>
i now was shown the holy soldiery
that in his blood he married jesus christ

but flying sees and sings the other which
<fail>
<fail>

Figure 13: Automatic translation of blind-test data from Dante's *Divine Comedy*.

6 Future Work

We plan to release all our of data in useful, processed form. Below we list directions for future research. In general, we see many interesting paths to pursue.

Analysis. Proper use of one-syllable words remains tricky. Lines coming out of generation and translation do not always scan naturally when read aloud by a person. We trace such errors to the fact that our lexical probabilities are context-independent. For example, we have:

```
P(S  | off) = 0.39
P(S* | off) = 0.61
```

When we look at Viterbi alignments from the analysis task, we see that when *off* is preceded by the word *far*, the probabilities reverse dramatically:

```
P(S  | off, after far) = 0.95
P(S* | off, after far) = 0.05
```

Similarly, the probability of stressing *at* is 40% in general, but this increases to 91% when the next word is *the*. Developing a model with context-dependent probabilities may be useful not only for improving generation and translation, but also for improving poetry analysis itself, as measured by anlaysis task accuracy.

Other potential improvements include the use of prior knowledge, for example, taking word length and spelling into account, and exploiting incomplete pronunciation dictionary information.

Generation. Evaluation is a big open problem for automatic poetry generation—even evaluating human poetry is difficult. Previous suggestions for automatic generation include acceptance for publication in some established venue, or passing the Turing test, i.e., confounding judges attempts to distinguish machine poetry from human poetry. The Turing test is currently difficult to pass with medium-sized Western poetry.

Translation. The advantage of translation over generation is that the source text provides a coherent sequence of propositions and images, allowing the machine to focus on "how to say" instead of "what to say." However, translation output lattices offer limited material to work with, and as we dig deeper into those lattices, we encounter increasingly disfluent ways to string together renderings of the source substrings.

An appealing future direction is to combine translation and generation. Rather than translating the source text, a program may instead use the source text for inspiration. Such a hybrid translation/generation program would not be bound to translate every word, but rather it could more freely combine lexical material from its translation tables with other grammatical and lexical resources. Interestingly, human translators sometimes work this way when they translate poetry—many excellent works have been produced by people with very little knowledge of the source language.

Paraphrasing. Recently, e→f translation tables have been composed with f→e tables, to make e→e tables that can paraphrase English into English (Bannard and Callison-Burch, 2005). This makes it possible to consider statistical translation of English prose into English poetry.

Acknowledgments

This work was partially supported by NSF grant IIS-0904684.

References

C. Bannard and C. Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proc. ACL*.

P. Brown, V. Della Pietra, S. Della Pietra, and R. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2).

B. Diaz-Agudo, P. Gervas, and P. A. Gonzalez-Calero. 2002. Poetry generation in COLIBRI. In *Proc. EC-CBR*.

P. Gervas. 2001. An expert system for the composition of formal Spanish poetry. *Journal of Knowledge-Based Systems*, 14:200–1.

P. Gervas. 2010. Engineering linguistic creativity: Bird flight and jet planes. Invited talk, CALC-10.

J. Graehl. 1997. Carmel finite-state toolkit. http://www.isi.edu/licensed-sw/carmel.

L. Jiang and M. Zhou. 2008. Generating Chinese couplets using a statistical MT approach. In *Proc. COLING*.

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 *Proc. ACL*.

H. Manurung, G. Ritchie, and H. Thompson. 2000. Towards a computational model of poetry generation. In *Proc. AISB'00 Symposium on Creative and Cultural Aspects and Applications of AI and Cognitive Science*.

H. Manurung. 2003. *An evolutionary algorithm approach to poetry generation*. Ph.D. thesis, University of Edinburgh.

Y. Netzer, D. Gabay, Y. Goldberg, and M. Elhadad. 2009. Gaiku : Generating Haiku with word associations norms. In *Proc. NAACL Workshop on Computational Approaches to Linguistic Creativity*.

N. Tosa, H. Obara, and M. Minoh. 2008. Hitch Haiku: An interactive supporting system for composing Haiku poem. In *Proc. International Conference on Entertainment Computing*.

M. T. Wong and A. H. W. Chun. 2008. Automatic Haiku generation using VSM. In *Proc. ACACOS*.

Discriminative Word Alignment with a Function Word Reordering Model

Hendra Setiawan
UMIACS
University of Maryland
hendra@umiacs.umd.edu

Chris Dyer
Language Technologies Institute
Carnegie Mellon University
cdyer@cs.cmu.edu

Philip Resnik
Linguistics and UMIACS
University of Maryland
resnik@umd.edu

Abstract

We address the modeling, parameter estimation and search challenges that arise from the introduction of reordering models that capture non-local reordering in alignment modeling. In particular, we introduce several reordering models that utilize (pairs of) function words as contexts for alignment reordering. To address the parameter estimation challenge, we propose to estimate these reordering models from a relatively small amount of manually-aligned corpora. To address the search challenge, we devise an iterative local search algorithm that stochastically explores reordering possibilities. By capturing non-local reordering phenomena, our proposed alignment model bears a closer resemblance to state-of-the-art translation model. Empirical results show significant improvements in alignment quality as well as in translation performance over baselines in a large-scale Chinese-English translation task.

1 Introduction

In many Statistical Machine Translation (SMT) systems, alignment represents an important piece of information, from which translation rules are learnt. However, while translation models have evolved from word-based to syntax-based modeling, the *de facto* alignment model remains word-based (Brown et al., 1993; Vogel et al., 1996). This gap between alignment modeling and translation modeling is clearly undesirable as it often generates tensions that would prevent the extraction of many useful translation rules (DeNero and Klein, 2007). Recent work, e.g. by Blunsom et al. (2009) and Haghihi et al. (2009) just to name a few, show that alignment models that bear closer resemblance to state-of-the-art translation model consistently yields not only a better alignment quality but also an improved translation quality.

In this paper, we follow this recent effort to narrow the gap between alignment model and translation model to improve translation quality. More concretely, we focus on the reordering component since we observe that the treatment of reordering remains significantly different when comparing alignment versus translation: the reordering component in state-of-the-art translation models has focused on long-distance reordering, but its counterpart in alignment models has remained focused on local reordering, typically modeling distortion based entirely on positional information. This leaves most alignment decisions to association-based scores.

Why is employing stronger reordering models more challenging in alignment than in translation? One answer can be attributed to the fact that alignment points are unobserved in parallel text, thus so are their reorderings. As such, introducing stronger reordering often further exacerbates the computational complexity to do inference over the model. Some recent alignment models appeal to external linguistic knowledge, mostly by using monolingual syntactic parses (Cherry and Lin, 2006; Pauls et al., 2010), which at the same time, provides an approximation of the bilingual syntactic divergences that drive the reordering. To our knowledge, however, this approach has been used mainly to constrain reordering possibilities, or to add to the generalization ability of association-based scores, not to directly model reordering in the context of alignment.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 534–544,
MIT, Massachusetts, USA, 9-11 October 2010.

In this paper, we introduce a new approach to improving the modeling of reordering in alignment. Instead of relying on monolingual parses, we condition our reordering model on the behavior of *function words* and the phrases that surround them. Function words are the “syntactic glue” of sentences, and in fact many syntacticians believe that functional categories, as opposed to substantive categories like noun and verb, are primarily responsible for cross-language syntactic variation (Ouhalla, 1991). Our reordering model can be seen as offering a reasonable approximation to more fully elaborated bilingual syntactic modeling, and this approximation is also highly practical, as it demands no external knowledge (other than a list of function words) and avoids the practical issues associated with the use of monolingual parses, e.g. whether the monolingual parser is robust enough to produce reliable output for every sentence in training data.

At a glance, our reordering model enumerates the function words on both source and target sides, modeling their reordering relative to their neighboring phrases, their neighboring function words, and the sentence boundaries. Because the frequency of function words is high, we find that by predicting the reordering of function words accurately, the reordering of the remaining words improves in accuracy as well. In total, we introduce six sub-models involving function words, and these serve as features in a log linear model. We train model weights discriminatively using Minimum Error Rate Training (MERT) (Och, 2003), optimizing F-measure.

The parameters of our sub-models are estimated from manually-aligned corpora, leading the reordering model more directly toward reproducing human alignments, rather than maximizing the likelihood of unaligned training data. This use of manual data for parameter estimation is a reasonable choice because these models depend on a small, fixed number of lexical items that occur frequently in language, hence only small training corpora are required. In addition, the availability of manually-aligned corpora has been growing steadily.

The remainder of the paper proceeds as follows. In Section 2, we provide empirical motivation for our approach. In Section 3, we discuss six submodels based on function word relationships and how their parameters are estimated; these are combined with additional features in Section 4 to produce a single discriminative alignment model. Section 5 describes a simple decoding algorithm to find the most probable alignment under the combined model, Section 6 describes the training of our discriminative model and Section 7 presents experimental results for the model using this algorithm. We wrap up in Sections 8 and 9 with a discussion of related work and a summary of our conclusions.

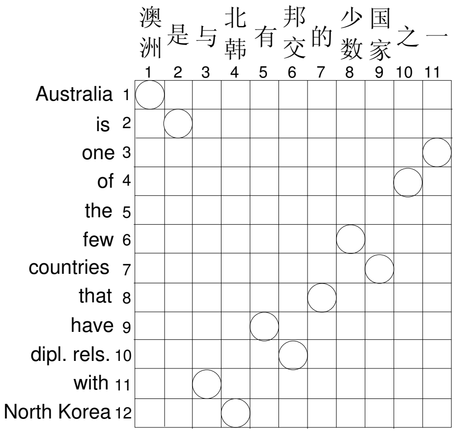


Figure 1: An aligned Chinese-English sentence pair.

2 Empirical Motivation

Fig. 1 shows an example of a Chinese-English sentence pair together with correct alignment points. Predicting the alignment for this particular Chinese-English sentence pair is challenging, since the significantly different syntactic structures of these two languages lead to non-monotone reordering. For example, an accurate alignment model should account for the fact that prepositional phrases in Chinese appear in a different order than in English, as illustrated by the movement of the phrase “与北韩/with North Korea” from the beginning of the Chinese noun phrase to the end of the corresponding English.

The central question that concerns us here is how to define and infer regularities that can be useful to predict alignment reorderings. The approach we take here is supported by empirical results from a pilot study, conducted as an inquiry into the idea of focusing on function words to model alignment reordering, which we briefly describe.

We took a Chinese-English manually-aligned corpus of approximately 21 thousand sentence pairs,

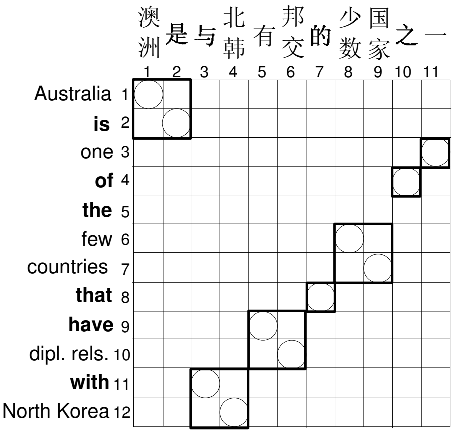


Figure 2: The all-monotone phrase pairs, indicated as rectangular areas in **bold**, that can be extracted from the Fig. 1 example.

and divided each sentence pair into *all-monotone phrase pairs*. Visually, an all-monotone phrase pair corresponds to a maximal block in the alignment matrix for which internal alignment points appear in monotone order from the top-left corner to the bottom-right corner. Fig. 2 illustrates seven such pairs that can be extracted from the example in Fig. 1. In total, there are 154,517 such phrase pairs in our manually-aligned corpus.

The alignment configuration internal to all-monotone phrase pair blocks is, obviously, monotonic, which is a configuration that is effectively modeled by traditional alignments models. On the other hand, the reordering between two adjacent blocks is the focus of our efforts since existing models are less effective at modeling non-monotonic alignment configurations. To measure the function words' potential to predict non-monotone reorderings, we examined the *border* words where two adjacent blocks meet. In particular, we are interested in how many adjacent blocks whose border words are function words.

The results of this pilot study were quite encouraging. If we consider only the Chinese side of the phrase pairs, 88.35% adjacent blocks have function words as their boundary words. If we consider only the English side, function words appear at the borders of 93.91% adjacent blocks. If we consider both the Chinese and English sides, the percentage increases to 95.53%. Notice that in Fig. 2, function words appear at the borders of *all* adjacent all-monotone phrase pairs, if both Chinese and English sides are considered. Clearly with such high coverage, function words are central in predicting non-monotone reordering in alignment.

3 Reordering with Function Words

The reordering models we describe follow our previous work using function word models for translation (Setiawan et al., 2007; Setiawan et al., 2009). The core hypothesis in this work is that function words provide robust clues to the reordering patterns of the phrases surrounding them. To make this insight useful for alignment, we develop features that score the alignment configuration of the neighboring phrases of a function word (which functions as an anchor) using two kinds of information: 1) the relative ordering of the phrases with respect to the function word anchor; and 2) the span of the phrases. This section provides a high level overview of our reordering model, which attempts to leverage this information.

To facilitate subsequent discussions, we introduce the notion of *monolingual* function word phrase FW_i, which consists of the tuple (Y_i, L_i, R_i), where Y_i is the i-th function word and L_i,R_i are its left and right neighboring phrases, respectively. Note that this notion of "phrase" is defined only for reordering purposes in our model, and does not necessarily correspond to a linguistic phrase. We define such phrases on both sides to cover as many non-monotone reorderings as possible, as suggested by the pilot study. To denote the side, we append a subscript: $FW_{i,S} = (Y_{i,S}, L_{i,S}, R_{i,S})$ refers to a function word phrase on the source side, and $FW_{i,T} = (Y_{i,T}, L_{i,T}, R_{i,T})$ to one on the target side. In our subsequent discussion, we will mainly use $FW_{i,S}$, and we will omit subscripts S or T if they are clear from context.

The primary objective of our reordering model is to predict the projection of monolingual function word phrases from one language to the other, inferring *bilingual* function word phrase pairs $FW_{i,S\rightarrow T} = (Y_{i,S\rightarrow T}, L_{i,S\rightarrow T}, R_{i,S\rightarrow T})$, which encode the two aforementioned pieces of information.[1] To infer these phrases, we take a probabilis-

[1] The subscript $S \rightarrow T$ denotes the projection direction from source to target. The subscript for the other direction is $T \rightarrow S$.

tic approach. For instance, to estimate the spans of $L_{i,S\rightarrow T}, R_{i,S\rightarrow T}$, our reordering model assumes that any span to the left of $Y_{i,S}$ is a possible $L_{i,S}$ and any span to the right of $Y_{i,S}$ is a possible $R_{i,S}$, deciding which is most probable via features, rather than committing to particular spans (e.g. as defined by a monolingual text chunker or parser). We only enforce one criterion on $L_{i,S\rightarrow T}$ and $R_{i,S\rightarrow T}$: they have to be the *maximal* alignment blocks satisfying the consistent heuristic (Och and Ney, 2004) that end or start with $Y_{i,S\rightarrow T}$ on the source S side respectively.[2]

To infer these phrases, we decompose $L_{i,S\rightarrow T}$ into $(o(L_{i,S\rightarrow T}),\ d(FW_{i-1,S\rightarrow T}),\ b(\langle s\rangle))$; similarly, $R_{i,S\rightarrow T}$ into $(o(R_{i,S\rightarrow T}),d(FW_{i+1,S\rightarrow T}),$ $b(\langle /s\rangle)$)). Taking the decomposition of $L_{i,S\rightarrow T}$ as a case in point, here $o(L_{i,S\rightarrow T})$ describes the reordering of the left neighbor $L_{i,S\rightarrow T}$ with respect to the function word $Y_{i,S\rightarrow T}$, while $d(FW_{i-1,S\rightarrow T})$ and $b(\langle s\rangle))$ probe the span of $L_{i,S\rightarrow T}$, i.e. whether it goes beyond the preceding function word phrase pairs $FW_{i-1,S\rightarrow T}$ and up to the beginning-of-sentence marker $\langle s\rangle$ respectively. The same definition applies to the decomposition of $R_{i,S\rightarrow T}$, where $FW_{i+1,S\rightarrow T}$ is the succeeding function word phrase pair and $\langle /s\rangle$ is the end-of-sentence marker.

3.1 Six (Sub-)Models

To model $o(L_{i,S\rightarrow T})$, $o(R_{i,S\rightarrow T})$, i.e. the reordering of the neighboring phrases of a function word, we employ the *orientation* model introduced by Setiawan et al. (2007). Formally, this model takes the form of probability distribution $P_{ori}(o(L_{i,S\rightarrow T}), o(R_{i,S\rightarrow T})|Y_{i,S\rightarrow T})$, which conditions the reordering on the lexical identity of the function word alignment (but independent of the lexical identity of its neighboring phrases). In particular, o maps the reordering into one of the following four orientation values (borrowed from Nagata et al. (2006)) with respect to the function word: Monotone Adjacent (MA), Monotone Gap (MG), Reverse Adjacent (RA) and Reverse Gap (RG). The Monotone/Reverse distinction indicates whether the projected order follows the original order, while the Adjacent/Gap distinction indicates whether the projections of the function word and the neighboring phrase are adjacent or separated by an intervening phrase.

To model $d(FW_{i-1,S\rightarrow T})$, $d(FW_{i+1,S\rightarrow T})$, i.e. whether $L_{i,S\rightarrow T}$ and $R_{i,S\rightarrow T}$ extend beyond the neighboring function word phrase pairs, we utilize the *pairwise dominance* model of Setiawan et al. (2009). Taking $d(FW_{i-1,S\rightarrow T})$ as a case in point, this model takes the form $P_{dom}(d(FW_{i-1,S\rightarrow T})|Y_{i-1,S\rightarrow T}, Y_{i,S\rightarrow T})$, where d takes one of the following four dominance values: leftFirst, rightFirst, dontCare, or neither. We will detail the exact formulation of these values in the next subsection. However, to provide intuition, the value of either leftFirst or neither for $d(FW_{i-1,S\rightarrow T})$ would suggest that the span of $L_{i,S\rightarrow T}$ doesn't extend to $Y_{i-1,S\rightarrow T}$; the further distinction between leftFirst and neither concerns with whether the span of $R_{i-1,S\rightarrow T}$ extends to $FW_{i,S\rightarrow T}$.

To model $b(\langle s\rangle)$, $b(\langle /s\rangle)$, i.e. whether the span of $L_{i,S\rightarrow T}$ and $R_{i,S\rightarrow T}$ extends up to sentence markers, we introduce the *borderwise dominance* model. Formally, this model is similar to the pairwise dominance model, except that we use the sentence boundaries as the anchors instead of the neighboring phrase pairs. This model captures longer distance dependencies compared to the previous two models; in the Chinese-English case, in particular, it is useful to discourage word alignments from crossing clause or sentence boundaries. The sentence boundary issue is especially important in machine translation (MT) experimentation, since the Chinese side of English-Chinese parallel text often includes long sentences that are composed of several independent clauses joined together; in such cases, words from one clause should be discouraged from aligning to words from other clauses. In Fig. 1, this model is potentially useful to discourage words from crossing the copula "是/is".

We define each model for all (pairs of) function word phrase pairs, forming features over a set of word alignments (A) between source (S) and target

[2] This heuristic is commonly used in learning phrase pairs from parallel text. The maximality ensures the uniqueness of L and R.

(T) sentence pair, as follows:

$$f_{ori} = \prod_{i=1}^{N} P_{ori}(o(L_i), o(R_i)|Y_i) \tag{1}$$

$$f_{dom} = \prod_{i=2}^{N} P_{dom}(d(FW_{i-1})|Y_{i-1}, Y_i) \tag{2}$$

$$f_{bdom} = \prod_{i=1}^{N} P_{bdom}(b(\langle s \rangle)|\langle s \rangle, Y_i) \cdot P_{bdom}(b(\langle /s \rangle)|Y_i, \langle /s \rangle) \tag{3}$$

where N is the number of function words (of the source side, in the $S \to T$ case). As the bilingual function word phrase pairs are uni-directional, we employ these three models in both directions, i.e. $T \to S$ as well as $S \to T$. As a result, there are six reordering models based on function words.

3.2 Prediction and Parameter Estimation

Given $FW_{i-1,S\to T}$ (and all other $FW_{\forall i'/i,S\to T}$), our reordering model has to decompose $L_{i,S\to T}$ into $(o(L_{i,S\to T}),\ d(FW_{i-1,S\to T}),\ b(\langle s \rangle))$; and $R_{i,S\to T}$ into $(o(R_{i,S\to T}), d(FW_{i+1,S\to T}),\ b(\langle /s \rangle)\ )$ during prediction and parameter estimation. In prediction mode (described in Section 5), it has to make the decomposition on the current state of alignment, while during parameter estimation, it has to make the same decomposition on the manually-aligned corpora. Since the process is identical, we proceed with the discussion in the context of parameter estimation, where the decomposition is performed to collect counts to estimate the parameters of our models.

Orientation model. Using $L_{i,S\to T}$ as a case in point and given ($Y_{i,S\to T}{=}s_l^l/t_m^m$, $L_{i,S\to T}{=}s_{l_1}^{l_2}/t_{m_1}^{m_2}$, $R_{i,S\to T}{=}s_{l_3}^{l_4}/t_{m_3}^{m_4}$)[3], the value of $o(L_{i,S\to T})$ in terms of Monotone/Reverse is:

$$\text{Monotone/Reverse} = \begin{cases} M, & m_2 < m, \\ R, & m < m_1. \end{cases} \tag{4}$$

while its value in terms of Adjacent/Gap values is:

$$\text{Adjacent/Gap} = \begin{cases} A, & |m - m_1| \vee |m - m_2| = 1, \\ G, & \text{otherwise.} \end{cases} \tag{5}$$

By adjusting the indices, the computation of $o(R_{i,S\to T})$ follows similarly to the procedure above.

Suppose we want to estimate the probability of $L_{i,S\to T}{=}MA$ for a particular Y_i. Note that here, we are interested in the lexical identity of Y_i, thus the index i is irrelevant. We first gather the counts of the orientation value for all $L_{i,S\to T}$ of Y_i in the corpus: $c(o(L_{i,S\to T}) \in \{MA, RA, MG, RG\}, Y_i)$. Then $P_{ori}(MA|Y_i)$ is estimated as follows:

$$P_{ori}(MA|Y_i) = \frac{c(MA, Y_i)}{c(Y_i)} \tag{6}$$

where $c(Y_i)$ is the frequency of Y_i in the corpus. The estimation of other orientation values as well as the $T \to S$ version of the model, follows the same procedure.

Pairwise and Borderwise dominance models. Given $R_{i,S\to T} = s_{l_1}^{l_2}/t_{m_1}^{m_2}$ and $L_{i+1,S\to T} = s_{l_3}^{l_4}/t_{m_3}^{m_4}$, i.e. the spans of the neighbors of a pair of neighboring function word phrase pairs ($Y_i = s_{l_5}^{l_5}/t_{m_5}^{m_5}, Y_{i+1} = s_{l_6}^{l_6}/t_{m_6}^{m_6}$), the value of $d(FW_{i+1,S\to T})$ is:

$$= \begin{cases} \textsf{leftFirst}, & l_2 \geq l_6 \bigwedge l_3 > l_5 \\ \textsf{rightFirst}, & l_2 < l_6 \bigwedge l_3 \leq l_5 \\ \textsf{dontCare}, & l_2 \geq l_6 \bigwedge l_3 \leq l_5 \\ \textsf{neither}, & l_2 < l_6 \bigwedge l_3 > l_5 \end{cases} \tag{7}$$

Note that the neighbors of the sentence markers for the borderwise models span the whole sentence, thus value of neither is impossible for these models.

Suppose we want to estimate the probability of Y_i and Y_{i+1} having a dontCare dominance value. Note that here we are interested in the lexical identity of Y_i and Y_{i+1}, thus the models are insensitive to the indices. We first gather the counts of the Y_i and Y_{i+1} having the dontCare value $c(\textsf{dontCare}, Y_i, Y_{i+1})$; then $P_{dom}(\textsf{dontCare}|Y_i, Y_{i+1})$ is estimated as follows:

$$P_{dom}(\textsf{dontCare}|Y_i, Y_{i+1}) = \frac{c(\textsf{dontCare}, Y_i, Y_{i+1})}{c(Y_i, Y_{i+1})} \tag{8}$$

where $c(Y_i, Y_{i+1})$ is the count of Y_i appears after Y_{i+1} in the training corpus without any other function word comes in between.

[3] We use subscripts to indicate the starting index, and superscripts the ending index.

4 Alignment Model

To use the function word alignment features described in the previous section to predict alignments, we use a linear model of the following form:

$$\hat{A} = \arg \max_{A \in \mathcal{A}(S,T)} \boldsymbol{\theta} \cdot \mathbf{f}(A, S, T) \qquad (9)$$

where $\mathcal{A}(S, T)$ is the set of all possible alignments of a source sentence S and target sentence T, and $\mathbf{f}(A, S, T)$ is a vector of feature functions on A, S, and T, and $\boldsymbol{\theta}$ is a parameter vector.

In addition to the six reordering models, our model employs several association-based scores that look at alignments in isolation. These features include:

1. Normalized log-likelihood ratio (LLR). This feature represents an association score, derived from statistical testing statistics. LLR (Dunning, 1993) has been widely used especially to measure lexical association. Since the values of LLR are unnormalized, we normalize them on a per-sentence basis, so that the normalized LLRs of, say, a particular source word to the target words in a particular sentence sum up to one.

2. Translation table from IBM model 4. This feature represents another association score, derived from a generative model, in particular the word-based IBM model 4. The use of this feature is widespread in recent alignment models, since it provides a relatively accurate initial prediction.

3. Translation table from manually-aligned corpora. This feature represents a gold-standard association score, based on human annotation. While attractive, this feature suffers from data sparseness issues since the lexical coverage of manually-aligned corpora, especially over content words, is very low. To overcome this issue, we design this feature to have two levels of granularity; as such, a fine-grained one is applied for function words and the coarse-grained one for content words.

4. Grow-diag-final alignments bonus. This feature encourages our alignment model to reuse alignment points that are part of the alignments created by the grow-diag-final heuristic, which we used as the baseline of our machine translation experiments.

5. Fertility model from IBM model 4. This feature, which is another by-product of IBM model 4, measures the probability of a certain word aligning to zero, one, or two or more words.

6. Null-alignment probability. This binomial feature models preference towards not aligning words, i.e. aligning to the NULL token. The intuition is to penalize NULL alignments depending on word class, by assigning lower probability mass to unaligned content words than to unaligned function words. In our experiment, we assign feature value 10^{-3} for a function word aligning to NULL, and 10^{-5} for a content word aligning to NULL.

Note that with the exception of the alignment bonus feature (4), all features are uni-directional, and therefore we employ these features in both directions just as was done for the reordering models.

5 Search

To find $\hat{A}$ using the model in Eq. 9, it is necessary to search $2^{|S|\times|T|}$ different alignment configurations, and, because of the non-local dependencies in some of our features, it is not possible to use dynamic programming to perform this search efficiently. We therefore employ an approximate search for the best alignment. We use a local search procedure which starts from some alignment (in our case, a symmetrized Model 4 alignment) and make local changes to it. Rather than taking a pure hill-climbing approach which greedily moves to locally better configurations (Brown et al., 1993), we use a stochastic search procedure which can move into lower-scoring states with some probability, similar to the Monte Carlo techniques used to draw samples from analytically intractable probability distributions.

5.1 Algorithm

To find $\hat{A}$, our search algorithm starts with an initial alignment $A^{(1)}$ and iteratively draws a new set by making a few small changes to the current set. For each step $i = [1, n]$, with alignment $A^{(i)}$, a set of neighboring alignments $\mathcal{N}(A^{(i)})$ is induced by applying small transformations (discussed below) to the current alignment. The next alignment $A^{(i+1)}$

is sampled from the following distribution:

$$p(A^{(i+1)}|S,T,A^{(i)}) = \frac{\exp \boldsymbol{\theta} \cdot \mathbf{f}(A^{(i+1)},S,T)}{Z(A^{(i)},S,T)}$$

where $Z(A^{(i)},S,T) = \sum_{A' \in \mathcal{N}(A^{(i)})} \exp \boldsymbol{\theta} \cdot \mathbf{f}(A',S,T)$

In addition to the current 'active' alignment configuration $A^{(i)}$, the algorithm keeps track of the highest scoring alignment observed so far, $A^{\max}$. After n steps, the algorithm returns $A^{\max}$ as its approximation of $\hat{A}$. In the experiments reported below, we initialized $A^{(1)}$ with the Model 4 alignments symmetrized by using the *grow-diag-final-and* heuristic (Koehn et al., 2003).

5.2 Alignment Neighborhoods

We now turn to a discussion of how the alignment neighborhoods used by our stochastic search algorithm are generated. We define three local transformation operations that apply to single columns of the alignment grid (which represent all of the alignments to the l^{th} source word), rows, or existing alignment points (l,m). Our three neighborhood generating operators are ALIGN, ALIGNEXCLUSIVE, and SWAP. The ALIGN operator applies to the l^{th} column of A and can either add an alignment point (l,m') or move an existing one (including to *null*, thus deleting it). ALIGNEXCLUSIVE adds an alignment point (l,m) and deletes all other points from row m. Finally, the SWAP operator swaps (l,m) and (l',m'), resulting in new alignment points (l,m') and (l',m). We increase the decoder's mobility by traversing the target side and applying the same steps above for each target word. Fig. 3 illustrates the three operators. By iterating over all columns l and rows m, the full alignment space $\mathcal{A}(S,T)$ can be explored.[4]

To further reduce the search space, an alignment point (l,m) is only admitted into a neighborhood if it is found in the high-recall alignment set $\mathcal{R}(S,T)$, which we define to be the model 4 union alignments (bidirectional model 4 symmetrized via union) plus the 5 best alignments according to the log-likelihood ratio.

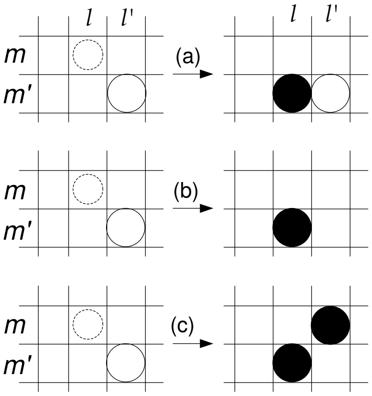


Figure 3: Illustrations for (a) ALIGN, (b) ALIGNEXCLUSIVE, and (c) SWAP operators, as applied to align the dotted, smaller circle (l,m) to (l,m'). The left hand side represents $\mathcal{A}^{(i)}$, while the right hand side represents a candidate for $\mathcal{A}^{(i+1)}$. The solid circles represent the new alignment points added to $\mathcal{A}^{(i+1)}$.

6 Discriminative Training

To set the model parameters $\boldsymbol{\theta}$, we used the minimum error rate training (MERT) algorithm (Och, 2003) to maximize the F-measure of the 1-best alignment of the model on a development set consisting of sentence pairs with manually generated alignments. The candidate set used by MERT to approximate the model is simply the set of alignments $\{A^{(1)}, A^{(2)}, \ldots, A^{(n)}\}$ encountered in the stochastic search.

While MERT does not scale to large numbers of features, the scarcity of manually aligned training data also means that models with large numbers of sparse features would be difficult to learn discriminatively, so this limitation is somewhat inherent in the problem space. Additionally, MERT has several advantages that make it particularly useful for our task. First, we can optimize F-measure of the alignments directly, which has been shown to correlate with translation quality in a downstream system (Fraser and Marcu, 2007b). Second, we are optimizing the quality of the 1-best alignments under the model. Since translation pipelines typically use only a single word alignment, this criterion is appropriate. Finally, and very importantly for us, MERT requires only an approximation of the model's hypothesis space to carry out optimization. Since we are using a stochastic search, this is crucial, since sub-

[4] Using only the ALIGN operator, it is possible to explore the full alignment space; however, using all three operators increases mobility.

sequent evaluations of the same sentence pair (even with the same weights) may result in a different candidate set.

Although MERT is a non-probabilistic optimizer, we explore the alignment space stochastically. This is necessary to make sure that the weights we use correspond to a probability distribution that is not overly peaked (which would result in a greedy hill-climbing search) or flat (which would explore the model space without information from the model). We found that normalizing the weights by the Euclidean norm resulted in a distribution that was well-balanced between the two extremes.

7 Experiments

We evaluated our proposed alignment model intrinsically on an alignment task and extrinsically on a large-scale translation task, focusing on Chinese-English as the language pair. Our training data consists of manually aligned corpora available from LDC (LDC2006E93 and LDC2008E57) and unaligned corpora, which include FBIS, ISI, HKNews and Xinhua. In total, the manually aligned corpora consist of more than 21 thousand sentence pairs, while the unaligned corpora consist of more than 710 thousand sentence pairs. The manually-aligned corpora are primarily used for training the reordering models and for discriminative training purposes. For translation experiments, we used cdec (Dyer et al., 2010), a fast implementation of hierarchical phrase-based translation models (Chiang, 2005), which represents a state-of-the-art translation system.

We constructed the list of function words in English manually and in Chinese from (Howard, 2002). Punctuation marks were added to the list, resulting in 883 and 359 tokens in the Chinese and English lists, respectively. For the alignment experiments, we took the first 500 sentence pairs from the newswire genre of the manually-aligned corpora and used the first 250 sentences as the development set, with the remaining 250 as the test set. To ensure blind experimentation, we excluded these sentence pairs from the training of the features, including the reordering models.

7.1 Alignment Quality

We used GIZA++, the implementation of the *de-facto* standard IBM alignment model, as our baseline alignment model. In particular, we used GIZA++ to align the concatenation of the development set, the test set, and the unaligned corpora, with 5, 5, 3 and 3 iterations of model 1, HMM, model 3, and model 4 respectively. Since the IBM model is asymmetric, we followed the standard practice of running GIZA++ twice, once in each direction, and combining the resulting outputs heuristically. We chose to use the grow-diag-final-and heuristic as it worked well for hierarchical phrase-based translation in our early experiments. We recorded the alignment quality of the test set as our baseline performance.

For our alignment model, we used the same set of training data. To align the test set, we first tuned the weights of the features in our discriminative alignment model using minimum error rate training (MERT) (Och, 2003) with $F_{\alpha=0.1}$ as the optimization criterion. At each iteration, our aligner outputs k-best alignments under current set of weights, from which MERT proceeds to compute the next set of weights. MERT terminates once the improvement over the previous iteration is lower than a predefined value. Once tuned, we ran our aligner on the test set and measured the quality of the resulting alignment as the performance of our model.

Model	P	R	$F_{0.5}$	$F_{0.1}$
gdfa	70.97	63.83	67.21	64.48
association	73.70	76.85	75.24	76.52
+ori	74.09	78.29	76.13	77.85
+dom	75.06	78.98	76.97	78.57
+bdom	**75.41**	**80.53**	**77.89**	**79.99**

Table 1: Alignment quality results ($F_{0.1}$) for our discriminative reordering models with various features (lines 2-5) versus the baseline IBM word-based Model 4 symmetrized using the grow-diag-final-and heuristic. The balanced $F_{0.5}$ measure is reported for reference. The best scores are **bolded**.

Table 1 reports the results of our experiments, which are conducted in an incremental fashion primarily to highlight the role of reordering modeling. The first line (gdfa) reports the baseline performance. In the first experiment (association), we employed only the association-based features described in Section 4. As shown, we obtain a significant improvement over baseline. This result is consistent with recent literature (Fraser and Marcu, 2007a) that shows that a discriminatively trained model outperforms baseline unsupervised models like GIZA++. In the second set of experiments, we added the reordering models into our discriminative model one by one, starting with the orientation models, then the pairwise dominance model and finally the borderwise dominance model, reported in lines +ori, +dom and +bdom respectively. As shown, each additional reordering model provides a significant additional improvement. The best result is obtained by employing all reordering models. These results empirically confirm our hypothesis that we can improve alignment quality by employing reordering models that capture non-local reordering phenomena.

7.2 Translation Quality

For translation experiments, we used the products from our intrinsic experiments to learn translation rules for the hierarchical phrase-based decoder, i.e. the features weights of the +bdom experiment to align the MT training data using our discriminative model. For our translation model, we used the standard features based on the relative frequency counts, including a 5-gram language model feature trained on the English portion of the whole training data plus portions of the Gigaword v2 corpus. Specifically, we tuned the weights of these features via MERT on the NIST MT06 set and we report the result on the NIST MT02, MT03, MT04 and MT05 sets.

	MT02	MT03	MT04	MT05
gdfa	25.61	32.05	31.80	29.34
this work	**26.56**	**33.79**	**32.61**	**30.47**

Table 2: The translation performance (BLEU) of hierarchical phrase-based translation trained on training data aligned by IBM model 4 symmetrized with the grow-diag-final-and heuristic, versus being trained on alignments by our discriminative alignment model. **Bolded** scores indicate that the improvement is statistically significant.

Table 2 shows the result of our translation experiments. In our alignment model, we employed the whole set of reordering models, i.e. the one reported in the +bdom line in Table 1. As shown, our discriminative alignment model produces a consistent and significant improvement over the baseline IBM model 4 ($p < 0.01$), ranging between 0.81 and 1.71 BLEU points.

8 Related Work

The focus of our work is to strengthen the reordering component of alignment modeling. Although the *de facto* standard, the IBM models do not generalize well in practice: the IBM approach employs a series of reordering models based on the word's position, but reordering depends on syntactic context rather than absolute position in the sentence. Over the years, there have been many proposals to improve these reordering models, most notably Vogel et al. (1996), which adds a first-order dependency. Nevertheless, the use of these distortion-based models remains widespread (Marcu and Wong, 2002; Moore, 2004).

Alignment modeling is challenging because it often has to consider a prohibitively large alignment space. Efforts to constrain the space generally comes from the use of Inversion Transduction Grammar (ITG) (Wu, 1997). Recent proposals that use ITG constraints include (Haghighi et al., 2009; Blunsom et al., 2009) just to name a few. More recent models have begun to use linguistically-motivated constraints, often in combination with ITG, primarily exploiting monolingual syntactic information (Burkett et al., 2010; Pauls et al., 2010).

Our reordering model is closely related to the model proposed by Zhang and Gildea (2005; 2006; 2007a), with respect to conditioning the reordering predictions on lexical items. These related models treat their lexical items as latent variables to be estimated from training data, while our model uses a fixed set of lexical items that correspond to the class of function words. With respect to the focus on function words, our reordering model is closely related to the UALIGN system (Hermjakob, 2009). However, UALIGN uses deep syntactic analysis and hand-crafted heuristics in its model.

9 Conclusions

Languages exhibit regularities of word order that are preserved when projected to another language. We use the notion of function words to infer such regularities, resulting in several reordering models that are employed as features in a discriminative alignment model. In particular, our models predict the reordering of function words by looking at their dependencies with respect to their neighboring phrases, their neighboring function words, and the sentence boundaries. By capturing such long-distance dependencies, our proposed alignment model contributes to the effort to unify alignment and translation. Our experiments demonstrate that our alignment approach achieves both its intrinsic and extrinsic goals.

Acknowledgements

This research was supported in part by the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-2-001. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the view of the sponsors.

References

Phil Blunsom, Trevor Cohn, Chris Dyer, and Miles Osborne. 2009. A gibbs sampler for phrasal synchronous grammar induction. In *ACL*, pages 782–790, Suntec, Singapore, August. Association for Computational Linguistics.

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.

David Burkett, John Blitzer, and Dan Klein. 2010. Joint parsing and alignment with weakly synchronized grammars. In *HLT-NAACL*, pages 127–135, Los Angeles, California, June. Association for Computational Linguistics.

Colin Cherry and Dekang Lin. 2006. Soft syntactic constraints for word alignment through discriminative training. In *COLING/ACL*, pages 105–112, Sydney, Australia, July. Association for Computational Linguistics.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *ACL*, pages 263–270, Ann Arbor, Michigan, June. Association for Computational Linguistics.

John DeNero and Dan Klein. 2007. Tailoring word alignments to syntactic machine translation. In *ACL*, pages 17–24, Prague, Czech Republic, June. Association for Computational Linguistics.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, March.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *ACL*, Uppsala, Sweden.

Alexander Fraser and Daniel Marcu. 2007a. Getting the structure right for word alignment: LEAF. In *EMNLP-CoNLL*, pages 51–60, Prague, Czech Republic, June. Association for Computational Linguistics.

Alexander Fraser and Daniel Marcu. 2007b. Measuring word alignment quality for statistical machine translation. *Computational Linguistics*, 33(3):293–303.

Aria Haghighi, John Blitzer, John DeNero, and Dan Klein. 2009. Better word alignments with supervised itg models. In *ACL*, pages 923–931, Suntec, Singapore, August. Association for Computational Linguistics.

Ulf Hermjakob. 2009. Improved word alignment with statistics and linguistic heuristics. In *EMNLP*, pages 229–237, Singapore, August. Association for Computational Linguistics.

Jiaying Howard. 2002. *A Student Handbook for Chinese Function Words*. The Chinese University Press.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *HTL-NAACL*, pages 127–133, Edmonton, Alberta, Canada, May. Association for Computational Linguistics.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *EMNLP*, July 23.

Robert C. Moore. 2004. Improving ibm word alignment model 1. In *ACL*, pages 518–525, Barcelona, Spain, July.

Masaaki Nagata, Kuniko Saito, Kazuhide Yamamoto, and Kazuteru Ohashi. 2006. A clustered global phrase reordering model for statistical machine translation. In *ACL*, pages 713–720, Sydney, Australia, July. Association for Computational Linguistics.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL*, pages 160–167.

Jamal Ouhalla. 1991. *Functional Categories and Parametric Variation*. Routledge.

Adam Pauls, Dan Klein, David Chiang, and Kevin Knight. 2010. Unsupervised syntactic alignment with inversion transduction grammars. In *HLT-NAACL*, pages 118–126, Los Angeles, California, June. Association for Computational Linguistics.

Hendra Setiawan, Min-Yen Kan, and Haizhou Li. 2007. Ordering phrases with function words. In *ACL*, pages 712–719, Prague, Czech Republic, June. Association for Computational Linguistics.

Hendra Setiawan, Min Yen Kan, Haizhou Li, and Philip Resnik. 2009. Topological ordering of function words in hierarchical phrase-based translation. In *ACL*, pages 324–332, Suntec, Singapore, August. Association for Computational Linguistics.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *COLING*, pages 836–841, Copenhagen.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404, Sep.

Hao Zhang and Daniel Gildea. 2005. Stochastic lexicalized inversion transduction grammar for alignment. In *ACL*. The Association for Computer Linguistics.

Hao Zhang and Daniel Gildea. 2006. Inducing word alignments with bilexical synchronous trees. In *ACL*. The Association for Computer Linguistics.

Hierarchical Phrase-based Translation Grammars Extracted from Alignment Posterior Probabilities

Adrià de Gispert, Juan Pino, William Byrne
Machine Intelligence Laboratory
Department of Engineering, University of Cambridge
Trumpington Street, CB2 1PZ, U.K.
{ad465|jmp84|wjb31}@eng.cam.ac.uk

Abstract

We report on investigations into hierarchical phrase-based translation grammars based on rules extracted from posterior distributions over alignments of the parallel text. Rather than restrict rule extraction to a single alignment, such as Viterbi, we instead extract rules based on posterior distributions provided by the HMM word-to-word alignment model. We define translation grammars progressively by adding classes of rules to a basic phrase-based system. We assess these grammars in terms of their expressive power, measured by their ability to align the parallel text from which their rules are extracted, and the quality of the translations they yield. In Chinese-to-English translation, we find that rule extraction from posteriors gives translation improvements. We also find that grammars with rules with only one nonterminal, when extracted from posteriors, can outperform more complex grammars extracted from Viterbi alignments. Finally, we show that the best way to exploit source-to-target and target-to-source alignment models is to build two separate systems and combine their output translation lattices.

1 Introduction

Current practice in hierarchical phrase-based translation extracts regular phrases and hierarchical rules from word-aligned parallel text. Alignment models estimated over the parallel text are used to generate these alignments, but these models are then typically used no further in rule extraction. This is less than ideal because these alignment models, even if they are not suitable for direct use in translation, can still provide a great deal of useful information beyond a single best estimate of the alignment of the parallel text. Our aim is to use alignment models to generate the statistics needed to build translation grammars. The challenge in doing so is to extend the current procedures, which are geared towards the use of a single alignment, to make more of what can be provided by alignment models. The goal is to extract a richer and more robust set of translation rules.

There are two aspects to hierarchical phrase-based translation grammars which concern us. The first is expressive power, which we take as the ability to generate known reference translations from sentences in the source language. This is determined by the degree of phrase movements and the translations allowed by the rules of the grammar. For a grammar with given types of rules, larger rule sets will yield greater expressive power. This motivates studies of grammars based on the rules which are extracted and the movement the grammar allows. The second aspect is of course translation accuracy. If the expressive power is adequate, then the desire is that the grammar assigns a high score to a correct translation.

We use posterior probabilities over parallel data to address both of these aspects. These posteriors allow us to build larger rule sets with improved translation accuracy. Ideally, for a sentence pair we wish to consider all possible alignments between all possible source and target phrases within these sentences. Given a grammar allowing certain types of movement, we would then extract all possible parses that are consistent with any alignments of these phrases.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 545–554,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

To make this approach feasible, we consider only phrase-to-phrase alignments with a high posterior probability under the alignment models. In this way, the alignment model probabilities guide rule extraction.

The paper is organized as follows. Section 2 reviews related work on using posteriors to extract phrases, as well as other approaches that tightly integrate word alignment and rule extraction. Section 3 describes rule extraction based on word and phrase posterior distributions provided by the HMM word-to-word alignment model. In Section 4 we define translation grammars progressively by adding classes of rules to a basic phrase-based system, motivating each rule type by the phrase movement it is intended to achieve. In Section 5 we assess these grammars in terms of their expressive power and the quality of the translations they yield in Chinese-to-English, showing that rule extraction from posteriors gives translation improvements. We also find that the best way to exploit source-to-target and target-to-source alignment models is to build two separate systems and combine their output translation lattices. Section 6 presents the main conclusions of this work.

2 Related Work

Some authors have previously addressed the limitation caused by decoupling word alignment models from grammar extraction. For instance Venugopal et al. (2008) extract rules from n-best lists of alignments for a syntax-augmented hierarchical system. Alignment n-best lists are also used in Liu et al. (2009) to create a structure called weighted alignment matrices that approximates word-to-word link posterior probabilities, from which phrases are extracted for a phrase-based system. Alignment posteriors have been used before for extracting phrases in non-hierarchical phrase-based translation (Venugopal et al., 2003; Kumar et al., 2007; Deng and Byrne, 2008).

In order to simplify hierarchical phrase-based grammars and make translation feasible with relatively large parallel corpora, some authors discuss the need for various filters during rule extraction (Chiang, 2007). In particular Lopez (2008) enforces a minimum span of two words per nonterminal, Zollmann et al. (2008) use a minimum count threshold for all rules, and Iglesias et al. (2009) propose a finer-grained filtering strategy based on rule patterns. Other approaches include insisting that target-side rules are well-formed dependency trees (Shen et al., 2008).

We also note approaches to tighter coupling between translation grammars and alignments. Marcu and Wong (2002) describe a joint-probability phrase-based model for alignment, but the approach is limited due to excessive complexity as Viterbi inference becomes NP-hard (DeNero and Klein, 2008). More recently, Saers et al. (2009) report improvement on a phrase-based system where word alignment has been trained with an inversion transduction grammar (ITG) rather than IBM models. Pauls et al. (2010) also use an ITG to directly align phrases to nodes in a string-to-tree model. Bayesian methods have been recently developed to induce a grammar directly from an unaligned parallel corpus (Blunsom et al., 2008; Blunsom et al., 2009). Finally, Cmejrek et al. (2009) extract rules directly from bilingual chart parses of the parallel corpus without using word alignments. We take a different approach in that we aim to start with very strong word alignment models and use them to guide grammar extraction.

3 Rule Extraction from Alignment Posteriors

The goal of rule extraction is to generate a set of good-quality translation rules from a parallel corpus. Rules are of the form $X{\rightarrow}\langle\gamma,\alpha,\sim\rangle$, where $\gamma, \alpha \in \{X \cup \mathbf{T}\}^+$ are the source and target sides of the rule, $\mathbf{T}$ denotes the set of terminals (words) and $\sim$ is a bijective function[1] relating source and target nonterminals X of each rule (Chiang, 2007). For each γ, the probability over translations α is set by relative frequency over the extracted examples from the corpus.

We take a general approach to rule extraction, as described by the following procedure. For simplicity we discuss the extraction of regular phrases, that is, rules of the form $X{\rightarrow}\langle w,w\rangle$, where $w \in \{\mathbf{T}\}^+$. Section 3.3 extends this procedure to rules with non-

[1] This function is defined if there are at least two nonterminals, and for clarity of presentation will be omitted in this paper

terminal symbols.

Given a sentence pair (f_1^J, e_1^I), the extraction algorithm traverses the source sentence and, for each sequence of terminals $f_{j_1}^{j_2}$, it considers all possible target-side sequences $e_{i_1}^{i_2}$ as translation candidates. Each target-side sequence that satisfies the alignment constraints $\mathcal{C}_A$ is ranked by the function f_R. For practical reasons, a set of selection criteria $\mathcal{C}_S$ is then applied to these ranked candidates and defines the set of translations of the source sequence that are extracted as rules. Each extracted rule is assigned a count f_C.

In this section we will explore variations of this rule extraction procedure involving alternative definitions of the ranking and counting functions, f_R and f_C, based on probabilities over alignment models.

Common practice (Koehn et al., 2003) takes a set of word alignment links $\mathbf{L}$ and defines the alignment constraints $\mathcal{C}_A$ so that there is a *consistency* between the links in the $(f_{j_1}^{j_2}, e_{i_1}^{i_2})$ phrase pair. This is expressed by $\forall (j,i) \in \mathbf{L} : (j \in [j_1, j_2] \wedge i \in [i_1, i_2]) \vee (j \notin [j_1, j_2] \;\wedge\; i \notin [i_1, i_2])$. If these constraints are met, then alignment probabilities are ignored and $f_R = f_C = 1$. We call this extraction Viterbi-based, as the set of alignment links is generally obtained after applying a symmetrization heuristic to source-to-target and target-to-source Viterbi alignments.

In the following section we depart from this approach and apply novel functions to rank and count target-side translations according to their quality in the context of each parallel sentence, as defined by the word alignment models. We also depart from common practice in that we do not use a set of links as alignment constraints. We thus find an increase in the number of extracted rules, and consequently better relative frequency estimates over translations.

3.1 Ranking and Counting Functions

We describe two alternative approaches to modify the functions f_R and f_C so that they incorporate the probabilities provided by the alignment models.

3.1.1 Word-to-word Alignment Posterior Probabilities

Word-to-word alignment posterior probabilities $p(l_{ji}|f_1^J, e_1^I)$ express how likely it is that the words in source position j and target position i are aligned given a sentence pair. These posteriors can be efficiently computed for Model 1, Model 2 and HMM, as described in (Brown et al., 1993; Venugopal et al., 2003; Deng and Byrne, 2008).

We will use these posteriors in functions to score phrase pairs. For a simple non-disjoint case $(f_{j_1}^{j_2}, e_{i_1}^{i_2})$ we use:

$$f_R(f_{j_1}^{j_2}, e_{i_1}^{i_2}) = \prod_{j=j_1}^{j_2} \sum_{i=i_1}^{i_2} \frac{p(l_{ji}|f_1^J, e_1^I)}{i_2 - i_1 + 1} \qquad (1)$$

which is very similar to the score used for lexical features in many systems (Koehn, 2010), with the link posteriors for the sentence pair playing the role of the Model 1 translation table.

For a particular source phrase, Equation 1 is not a proper conditional probability distribution over all phrases in the target sentence. Therefore it cannot be used as such without further normalization. Indeed we find that this distribution is too sharp and overemphasises short phrases, so we use $f_C = 1$. However, it does allow us to rank target phrases as possible translations. In contrast to the common extraction procedure described in the previous section, the ranking approach described here can lead to a much more exhaustive extraction unless selection criteria are applied. These we describe in Section 3.2.

We note that Equation 1 can be computed using link posteriors provided by alignment models trained on either source-to-target or target-to-source translation directions.

3.1.2 Phrase-to-phrase Alignment Posterior Probabilities

Rather than limit ourselves to word-to-word link posteriors we can define alignment probability distributions over phrase alignments. We do this by defining the set of alignments A as $A(j_1, j_2; i_1, i_2) = \{a_1^J : a_j \in [i_1, i_2] \text{ iff } j \in [j_1, j_2]\}$, where a_j is the random process that describes word-to-word alignments. These are the alignments from which the phrase pair $(f_{j_1}^{j_2}, e_{i_1}^{i_2})$ would be extracted.

The posterior probability of these alignments given the sentence pair is defined as follows:

$$p(A|e_1^I, f_1^J) = \frac{\sum_{a_1^J \in A} p(f_1^J, a_1^J|e_1^I)}{\sum_{a_1^J} p(f_1^J, a_1^J|e_1^I)} \qquad (2)$$

G_0	G_1	G_2	G_3
$S\rightarrow\langle X,X\rangle$ $S\rightarrow\langle S\ X,S\ X\rangle$ $X\rightarrow\langle w,w\rangle$	$X\rightarrow\langle w\ X,X\ w\rangle$ $X\rightarrow\langle X\ w,w\ X\rangle$	$X\rightarrow\langle w\ X,X\ w\rangle$ $X\rightarrow\langle X\ w,w\ X\rangle$ $X\rightarrow\langle w\ X,w\ X\rangle$	$X\rightarrow\langle w\ X,X\ w\rangle$ $X\rightarrow\langle X\ w,w\ X\rangle$ $X\rightarrow\langle w\ X,w\ X\rangle$ $X\rightarrow\langle w\ X\ w,w\ X\ w\rangle$

Table 1: Hierarchical phrase-based grammars containing different types of rules. The grammar expressivity is greater as more types of rules are included. In addition to the rules shown in the respective columns, G_1, G_2 and G_3 also contain the rules of G_0.

With IBM models 1 and 2, the numerator and denominator in Equation 2 can be computed in terms of posterior link probabilities (Deng, 2005). With the HMM model, the denominator is computed using the forward algorithm while the numerator can be computed using a modified forward algorithm (Deng, 2005).

These phrase posteriors directly define a probability distribution over the alignments of translation candidates, so we use them both for ranking and scoring extracted rules, that is $f_R = f_C = p$. This approach assigns a fractional count to each extracted rule, which allows finer estimation of the forward and backward translation probability distributions.

3.2 Alignment Constraints and Selection Criteria

In order to keep this process computationally tractable, some extraction constraints are needed. In order to extract a phrase pair $(f_{j_1}^{j_2}, e_{i_1}^{i_2})$, we define the following:

- $\mathcal{C}_A$ requires at least one pair of positions (j,i) : $(j \in [j_1,j_2] \wedge i \in [i_1,i_2])$ with word-to-word link posterior probability $p(l_{ji}|f_1^J, e_1^I) > 0.5$, and that there is no pair of positions (j,i) : $(j \in [j_1,j_2] \wedge i \notin [i_1,i_2]) \vee (j \notin [j_1,j_2] \wedge i \in [i_1,i_2])$ with $p(l_{ji}|f_1^J, e_1^I) > 0.5$

- $\mathcal{C}_S$ allows only the k best translation candidates to be extracted. We use $k = 3$ for regular phrases, and $k = 2$ for hierarchical rules.

Note that we do not discard rules according to their scores f_C at this point (unlike Liu et al. (2009)), since we prefer to add all phrases from all sentence pairs before carrying out such filtering steps.

Once all rules over the entire collection of parallel sentences have been extracted, we require each rule to occur at least n_{obs} times and with a forward translation probability $p(\alpha|\gamma) > 0.01$ to be used for translation.

3.3 Extraction of Rules with Nonterminals

Extending the procedure previously described to the case of more complex hierarchical rules including one or even two nonterminals is conceptually straightforward. It merely requires that we traverse the source and target sentences and consider possibly disjoint phrase pairs. Optionally, the alignment constraints can also be extended to apply on the nonterminal X.

Equation 1 is then only modified in the limits of the product and summation, whereas Equation 2 remains unchanged, as long as the set of valid alignments A is redefined. For example, for a rule of the form $X\rightarrow\langle w\ X\ w,w\ X\ w\rangle$, we use $A \equiv A(j_1, j_2; j_3, j_4; i_1, i_2; i_3, i_4)$.

4 Hierarchical Translation Grammar Definition

In this section we define the hierarchical phrase-based synchronous grammars we use for translation experiments. Each grammar is defined by the type of hierarchical rules it contains. The rule type can be obtained by replacing every sequence of terminals by a single symbol 'w', thus ignoring the identity of the words, but capturing its generalized structure and the kind of reordering it encodes (this was defined as rule pattern in Iglesias et al. (2009)).

A monotonic phrase-based translation grammar G_0 can be defined as shown in the left-most column of Table 1; it includes all regular phrases, represented by the rule type $X\rightarrow\langle w,w\rangle$, and the two glue

(G_0) R^1: $S\rightarrow\langle X,X\rangle$
(G_0) R^2: $X\rightarrow\langle \mathrm{s}_2\ \mathrm{s}_3,\mathrm{t}_2\rangle$
(G_1) R^3: $X\rightarrow\langle \mathrm{s}_1\ X,X\ \mathrm{t}_3\rangle$
(G_1) R^4: $X\rightarrow\langle X\ \mathrm{s}_4,\mathrm{t}_1\ X\rangle$
(G_2) R^5: $X\rightarrow\langle \mathrm{s}_1\ X,\mathrm{t}_7\ X\rangle$
(G_3) R^6: $X\rightarrow\langle \mathrm{s}_1\ X\ \mathrm{s}_4,\mathrm{t}_5\ X\ \mathrm{t}_6\rangle$

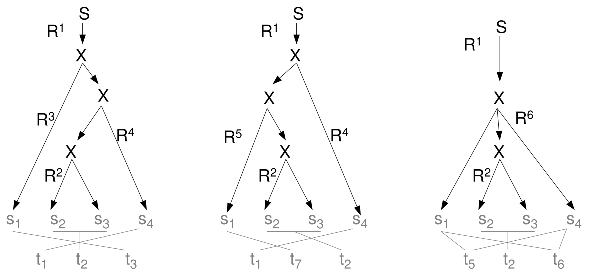


Figure 1: Example of a hierarchical translation grammar and two parsing trees following alternative rule derivations for the input sentence $s_1 s_2 s_3 s_4$.

rules that allow concatenation. Our approach is now simple: we extend this grammar by successively incorporating sets of hierarchical rules. The goal is to obtain a grammar with few rule types but which is capable of generating a rich set of translation candidates for a given input sentence.

With this in mind, we define the following three grammars, also summarized in Table 1:

- $G_1 := G_0 \bigcup \{\ X\rightarrow\langle w\ X,X\ w\rangle\ ,\ X\rightarrow\langle X\ w,w\ X\rangle\ \}$. This incorporates reordering capabilities with two rule types that place the unique nonterminal in an opposite position in each language; we call these 'phrase swap rules'. Since all nonterminals are of the same category X, nested reordering is possible. However, this needs to happen consecutively, *i.e.* a swap must apply after a swap, or the rule is concatenated with the glue rule.

- $G_2\ :=\ G_1 \bigcup \{\ X\rightarrow\langle w\ \ X,w\ \ X\rangle\ \}$. This adds monotonic concatenation capabilities to the previous translation grammar. The glue rule already allows rule concatenation. However, it does so at the S category, that is, it concatenates phrases and rules *after* they have been reordered, in order to complete a sentence. With this new rule type, G_2 allows phrase/rule concatenation *before* reordering with another hierarchical rule. Therefore, nested reordering does not require successive swaps anymore.

- $G_3\ :=\ G_2 \bigcup \{\ X\rightarrow\langle w\ X\ w,w\ X\ w\rangle\ \}$. This adds single nonterminal rules with disjoint terminal sequences, which can encode a monotonic or reordered relationship between them, depending on what their alignment was in the parallel corpus. Although one could expect the movement captured by this phrase-disjoint rule type to be also present in G_2 (via two swaps or one concatenation plus one swap), the terminal sequences w may differ.

Figure 1 shows an example set of rules indicating to which of the previous grammars each rule belongs, and shows three translation candidates as generated by grammars G_1 (left-most tree), G_2 (middle tree) and G_3 (right-most tree). Note that the middle tree cannot be generated with G_1 as it requires monotonic concatenation before reordering with rule R^4.

The more rule types a hierarchical grammar contains, the more different rule derivations and the greater the search space of alternative translation candidates. This is also connected to how many rules are extracted per rule type. Ideally we would like the grammar to be able to generate the correct translation of a given input sentence, without overgenerating too many other candidates, as that makes the translation task more difficult.

We will make use of the parallel data in measuring the ability of a grammar to generate correct translations. By extracting rules from a parallel sentence, we translate them and observe whether the translation grammar is able to produce the parallel target translation. In Section 5.1 we evaluate this for a Chinese-to-English task.

4.1 Reducing Grammar Redundancy

Let us discuss grammar G_2 in more detail. As described in the previous section, the motivation for including rule type $X \rightarrow \langle w\ X, w\ X \rangle$ is that the grammar be able to carry out monotonic concatenation *before* applying another hierarchical rule with reordering. This movement is permitted by this rule type, but the use of a single nonterminal category X also allows the grammar to apply the concatenation *after* reordering, that is, immediately before the glue rule is applied. This creates significant redundancy in rule derivations, as this rule type is allowed to act as a glue rule. For example, given an input sentence $s_1 s_2$ and the following simple grammar:

R^0: $S \rightarrow \langle X, X \rangle$
R^1: $S \rightarrow \langle S\ X, S\ X \rangle$
R^2: $X \rightarrow \langle \mathrm{s}_1, \mathrm{t}_1 \rangle$
R^3: $X \rightarrow \langle \mathrm{s}_2, \mathrm{t}_2 \rangle$
R^4: $X \rightarrow \langle \mathrm{s}_1\ X, \mathrm{t}_1\ X \rangle$

two derivations are possible: R^2,R^0,R^3,R^1 and R^3,R^4,R^0, and the translation result is identical.

To avoid this situation we introduce a nonterminal M in the left-hand side of monotonic concatenation rules of G_2. All rules are allowed to use nonterminals X and M in their right-hand side, except the glue rules, which can only take X. In the context of our example, R^4 is substituted by:

R^{4a}: $M \rightarrow \langle \mathrm{s}_1\ X, \mathrm{t}_1\ X \rangle$
R^{4b}: $M \rightarrow \langle \mathrm{s}_1\ M, \mathrm{t}_1\ M \rangle$

so that only the first derivation is possible: R^2,R^0,R^3,R^1, because applying R^3,R^{4a} yields a nonterminal M that cannot be taken by the glue rule R^0.

5 Experiments

We report experiments in Chinese-to-English translation. Our system is trained on a subset of the GALE 2008 evaluation parallel text;[2] this is approximately 50M words per language. We report translation results on a development set *tune-nw* and a test set *test-nw1*. These contain translations produced by the GALE program and portions of the newswire sections of MT02 through MT06. They contain 1,755 sentences and 1,671 sentences respectively. Results are also reported on a smaller held-out test set *test-nw2*, containing 60% of the NIST newswire portion of MT06, that is, 369 sentences.

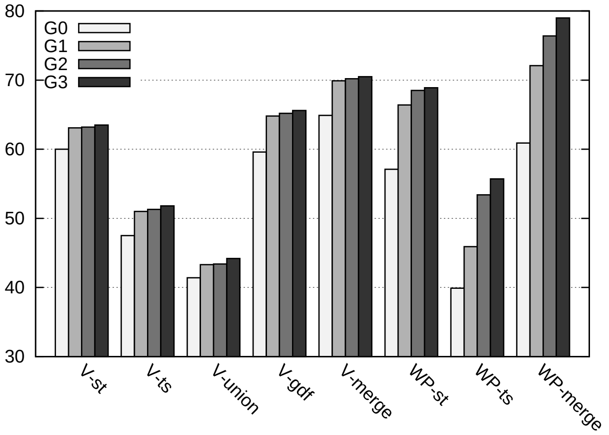


Figure 2: Percentage of parallel sentences successfully aligned for various extraction methods and grammars.

The parallel texts for both language pairs are aligned using MTTK (Deng and Byrne, 2008). For decoding we use HiFST, a lattice-based decoder implemented with Weighted Finite State Transducers (de Gispert et al., 2010). Likelihood-based search pruning is applied if the number of states in the lattice associated with each CYK grid cell exceeds 10,000, otherwise the entire search space is explored. The language model is a 4-gram language model estimated over the English side of the parallel text and the AFP and Xinhua portions of the English Gigaword Fourth Edition (LDC2009T13), interpolated with a zero-cutoff stupid-backoff (Brants et al., 2007) 5-gram estimated using 6.6B words of English newswire text. In tuning the systems, standard MERT (Och, 2003) iterative parameter estimation under IBM BLEU[3] is performed on the development sets.

5.1 Measuring Expressive Power

We measure the expressive power of the grammars described in the previous section by running the translation system in alignment mode (de Gispert et al., 2010) over the parallel corpus. Conceptually, this is equivalent to replacing the language model by the target sentence and seeing if the system is able to find any candidate. Here the weights assigned to the

[2] See http://projects.ldc.upenn.edu/gale/data/catalog.html. We excluded the UN material and the LDC2002E18, LDC2004T08, LDC2007E08 and CUDonga collections.

[3] See ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v13.pl

Grammar	Extraction	# Rules	*tune-nw*			*test-nw1*	*test-nw2*
			time	*prune*	BLEU	BLEU	BLEU
G_H	**V-union**	979149	3.7	0.3	35.1	35.6	37.6
G_1	**V-union**	613962	0.4	0.0	33.6	34.6	36.4
	WP-st	920183	0.9	0.0	34.3	34.8	37.5
	PP-st	893542	1.4	0.0	34.4	35.1	37.7
G_2	**V-union**	734994	1.0	0.0	34.5	35.4	37.2
	WP-st	1132386	5.8	0.5	35.1	36.0	37.7
	PP-st	1238235	7.8	0.7	35.5	36.4	38.2
G_3	**V-union**	966828	1.2	0.0	34.9	35.3	37.0
	WP-st	2680712	8.3	1.1	35.1	36.2	37.9
	PP-st	5002168	10.7	2.6	35.5	36.4	38.5

Table 2: Chinese-to-English translation results with alternative grammars and extraction methods (lower-cased BLEU shown). Time (secs/word) and prune (times/word) measurements done on *tune-nw* set.

rules are irrelevant, as only the ability of the grammar to create a desired hypothesis is important.

We compare the percentage of target sentences that can be successfully produced by grammars G_0, G_1, G_2 and G_3 for the following extraction methods:

- **Viterbi (V)**. This is the standard extraction method based on a set of alignment links. We distinguish four cases, depending on the model used to obtain the set of links: source-to-target (**V-st**), target-to-source (**V-ts**), and two common symmetrization strategies: union (**V-union**) and grow-diag-final (**V-gdf**), described in (Koehn et al., 2003).

- **Word Posteriors (WP)**. The extraction method is based on word alignment posteriors described in Section 3.1.1. These rules can be obtained either from the posteriors of the source-to-target (**WP-st**) or the target-to-source (**WP-ts**) alignment models. We apply the alignment constraints and selection criteria described in Section 3.2. We do not report alignment percentages when using phrase posteriors (as described in Section 3.1.2) as they are roughly identical to the **WP** case.

- Finally, in both cases, we also report results when merging the extracted rules in both directions into a single rule set (**V-merge** and **WP-merge**).

Figure 2 shows the results obtained for a random selection of 10,000 parallel corpus sentences. As expected, we can see that for any extraction method, the percentage of aligned sentences increases when switching from G_0 to G_1, G_2 and G_3. Posterior-based extraction is shown to outperform standard methods based on a Viterbi set of alignments for nearly all grammars. The highest alignment percentages are obtained when merging rules obtained under models trained in each direction (**WP-merge**), approximately reaching 80% for grammar G_3.

The maximum rule span in alignment was allowed to be 15 words, so as to be similar to translation, where the maximum rule span is 10 words. Relaxing this in alignment to 30 words yields approximately 90% coverage for **WP-merge** under G_3.

We note that if alignment constraints $\mathcal{C}_A$ and selection criteria $\mathcal{C}_S$ were not applied, that is $k = \infty$, then alignment percentages would be 100% even for G_0, but the extracted grammar would include many noisy rules with poor generalization power and would suffer from overgeneration.

5.2 Translation Results

In this section we investigate the translation performance of each hierarchical grammar, as defined by rules obtained from three rule extraction methods:

- **Viterbi union (V-union)**. Standard rule extraction from the union of the source-to-target and target-to-source alignment link sets.

- **Word Posteriors (WP-st)**. Extraction based on word posteriors as described in Section 3.1.1. The posteriors are provided by the source-to-target alignment model. Alignment constraints and selection criteria of Section 3.2 are applied, with $n_{obs} = 2$.

- **Phrase Posteriors (PP-st)**. Extraction based on phrase alignment posteriors, as described in Section 3.1.2, with fractional counts proportional to the phrase probability under the source-to-target alignment model. Alignment constraints and selection criteria of Section 3.2 are applied, with $n_{obs} = 0.2$.

Table 2 reports the translation results, as well as the number of extracted rules in each case. It also shows the following decoding statistics as measured on the *tune-nw* set: decoding time in seconds per input word, and number of instances of search pruning (described in Section 5) per input word.

As a contrast, we extract rules according to the heuristics introduced in (Chiang, 2007) and apply the filters described in (Iglesias et al., 2009) to generate a standard hierarchical phrase-based grammar G_H. This uses rules with up to two nonadjacent nonterminals, but excludes identical rule types such as $X \rightarrow \langle w\ X, w\ X \rangle$ or $X \rightarrow \langle w\ X_1\ w\ X_2, w\ X_1\ w\ X_2 \rangle$, which were reported to cause computational difficulties without a clear improvement in translation (Iglesias et al., 2009).

Grammar expressivity. As expected, for the standard extraction method (see rows entitled **V-union**), grammar G_1 is shown to underperform all other grammars due to its structural limitations. On the other hand, grammar G_2 obtains much better scores, nearly generating the same translation quality as the baseline grammar G_H. Finally, G_3 does not prove able to outperform G_2, which suggests that the phrase-disjoint rules with one nonterminal are redundant for the translation grammar.

Rule extraction method. For all grammars, we find that the proposed extraction methods based on alignment posteriors outperform standard Viterbi-based extraction, with improvements ranging from 0.5 to 1.1 BLEU points for *test-nw1* (depending on the grammar) and from 1.0 to 1.5 for *test-nw2*. In all cases, the use of phrase posteriors **PP** is the best option. Interestingly, we find that G_2 extracted with **WP** and **PP** methods outperforms the more complex G_H grammar as obtained from Viterbi alignments.

Rule set statistics. For grammar G_2 evaluated on the *tune-nw* set, standard Viterbi-based extraction produces 0.7M rules, whereas the WP and PP extraction methods yield 1.1M and 1.2M rules respectively. We further analyse the sets of rules $X \rightarrow \langle \gamma, \alpha, \sim \rangle$ in terms of the number of distinct source and target sequences γ and α which are extracted. Viterbi extraction yields 82k distinct source sequences whereas the WP and PP methods yield 116k and 146k sequences, respectively. In terms of the average number of target sequences for each source sequence, Viterbi extraction yields an average of 8.7 while WP and PP yield 9.7 and 8.4 rules on average. This shows that method **PP** yields wider coverage but with sharper forward rule translation probability distributions than method **WP**, as the average number of translations per rule is determined by the $p(\alpha|\gamma) > 0.01$ threshold mentioned in Section 3.2.

Decoding time and pruning in search. In connection to the previous comments, we find an increased need for search pruning, and subsequently slower decoding speed, as the search space grows larger with methods **WP** and **PP**. A larger search space is created by the larger rule sets, which allows the system to generate new hypotheses of better quality.

5.3 Rule Concatenation in Grammar G_2

In Section 4.1 we described a strategy to reduce grammar redundancy by introducing an additional nonterminal M for monotonic concatenation rules. We find that without this distinction among nonterminals, search pruning and decoding time are increased by a factor of 1.5, and there is a slight degradation in BLEU ($\sim$0.2) as more search errors are introduced.

Another relevant aspect of this grammar is the actual rule type selected for monotonic concatenation. We described using type $X \rightarrow \langle w\ X, w\ X \rangle$ (concatenation on the right), but one could also include $X \rightarrow \langle X\ w, X\ w \rangle$ (concatenation on the left), or both, for the same purpose. We evaluated the three alternatives and found that scores are identical when either including right or left concatenation types, but including both is harmful for performance, as the need to prune and decoding time increase by a factor of 5 and 4, respectively, and we observe again a slight degradation in performance.

Rule Extraction	*tune-nw*	*test-nw1*	*test-nw2*
V-st	34.7	35.6	37.5
V-ts	34.0	34.8	36.6
V-union	34.5	35.4	37.2
V-gdf	34.4	35.3	37.1
WP-st	35.1	36.0	37.7
WP-ts	34.5	35.0	37.0
PP-st	35.5	36.4	38.2
PP-ts	34.8	35.3	37.2
PP-merge	35.5	36.4	38.4
PP-merge-MERT	35.5	36.4	38.3
LMBR(**V-st)**	35.0	35.8	38.4
LMBR(**V-st,V-ts)**	35.5	36.3	38.9
LMBR(**PP-st)**	36.1	36.8	38.8
LMBR(**PP-st,PP-ts)**	36.4	36.9	39.3

Table 3: Translation results under grammar G_2 with individual rule sets, merged rule sets, and rescoring and system combination with lattice-based MBR (lower-cased BLEU shown)

5.4 Symmetrizing Alignments of Parallel Text

In this section we investigate extraction from alignments (and posterior distributions) over parallel text which are generated using alignment models trained in the source-to-target (**st**) and target-to-source (**ts**) directions. Our motivation is that symmetrization strategies have been reported to be beneficial for Viterbi extraction methods (Och and Ney, 2003; Koehn et al., 2003).

Results are shown in Table 3 for grammar G_2. We find that rules extracted under the source-to-target alignment models (**V-st**, **WP-st** and **PP-st**) consistently perform better than the **V-ts**, **WP-ts** and **PP-ts** cases. Also, for Viterbi extraction we find that the source-to-target **V-st** case performs better than any of the symmetrization strategies, which contradicts previous findings for non-hierarchical phrase-based systems(Koehn et al., 2003).

We use the **PP** rule extraction method to extract two sets of rules, under the **st** and **ts** alignment models respectively. We now investigate two ways of merging these sets into a single grammar for translation. The first strategy is **PP-merge** and merges both rule sets by assigning to each rule the maximum count assigned by either alignment model. We then extend the previous strategy by adding three binary feature functions to the system, indicating whether the rule was extracted under the '**st**' model, the '**ts**' model or both. The motivation is that MERT can weight rules differently according to the alignment model they were extracted from. However, we do not find any improvement with either strategy.

Finally, we use linearised lattice minimum Bayes-risk decoding (Tromble et al., 2008; Blackwood et al., 2010) to combine translation lattices (de Gispert et al., 2010) as produced by rules extracted under each alignment direction (see rows named LMBR(**V-st,V-ts)**) and LMBR(**PP-st,PP-ts)**). Gains are consistent when comparing this to applying LMBR to each of the best individual systems (rows named LMBR(**V-st)** and LMBR(**PP-st)**). Overall, the best-performing strategy is to extract two sets of translation rules under the phrase pair posteriors in each translation direction, and then to perform translation twice and merge the results.

6 Conclusion

Rule extraction based on alignment posterior probabilities can generate larger rule sets. This results in grammars with more expressive power, as measured by the ability to align parallel sentences. Assigning counts equal to phrase posteriors produces better estimation of rule translation probabilities. This results in improved translation scores as the search space grows.

This more exhaustive rule extraction method permits a grammar simplification, as expressed by the phrase movement allowed by its rules. In particular a simple grammar with rules of only one nonterminal is shown to outperform a more complex grammar built on rules extracted from Viterbi alignments. Finally, we find that the best way to exploit alignment models trained in each translation direction is to extract two rule sets based on alignment posteriors, translate the input independently with each rule set and combine translation output lattices.

Acknowledgments

This work was supported in part by the GALE program of the Defense Advanced Research Projects

Agency, Contract No. HR0011- 06-C-0022.

References

Graeme Blackwood, Adrià de Gispert, and William Byrne. 2010. Efficient Path Counting Transducers for Minimum Bayes-Risk Decoding of Statistical Machine Translation Lattices. In *Proceedings of ACL, Short Papers*, pages 27–32.

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. Bayesian Synchronous Grammar Induction. In *Advances in Neural Information Processing Systems*, volume 21, pages 161–168.

Phil Blunsom, Trevor Cohn, Chris Dyer, and Miles Osborne. 2009. A gibbs sampler for phrasal synchronous grammar induction. In *Proceedings of the ACL*, pages 782–790.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of EMNLP-ACL*, pages 858–867.

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

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Martin Cmejrek, Bowen Zhou, and Bing Xiang. 2009. Enriching SCFG Rules Directly From Efficient Bilingual Chart Parsing. In *Proceedings of IWSLT*, pages 136–143.

Adrià de Gispert, Gonzalo Iglesias, Graeme Blackwood, Eduardo R. Banga, and William Byrne. 2010. Hierarchical phrase-based translation with weighted finite state transducers and shallow-n grammars. *Computational Linguistics*, 36(3).

John DeNero and Dan Klein. 2008. The complexity of phrase alignment problems. In *Proceedings of ACL-HLT, Short Papers*, pages 25–28.

Yonggang Deng and William Byrne. 2008. HMM word and phrase alignment for statistical machine translation. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(3):494–507.

Yonggang Deng. 2005. *Bitext Alignment for Statistical Machine Translation*. Ph.D. thesis, Johns Hopkins University.

Gonzalo Iglesias, Adrià de Gispert, Eduardo R. Banga, and William Byrne. 2009. Rule filtering by pattern for efficient hierarchical translation. In *Proceedings of the EACL*, pages 380–388.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT/NAACL*, pages 48–54.

Philipp Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press.

Shankar Kumar, Franz J. Och, and Wolfgang Macherey. 2007. Improving word alignment with bridge languages. In *Proceedings of EMNLP-CoNLL*, pages 42–50.

Yang Liu, Tian Xia, Xinyan Xiao, and Qun Liu. 2009. Weighted alignment matrices for statistical machine translation. In *Proceedings of EMNLP*, pages 1017–1026.

Adam Lopez. 2008. Tera-scale translation models via pattern matching. In *Proceedings of COLING*, pages 505–512.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of EMNLP*, pages 133–139.

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

Franz J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167.

Adam Pauls, Dan Klein, David Chiang, and Kevin Knight. 2010. Unsupervised syntactic alignment with inversion transduction grammars. In *Proceedings of the HLT-NAACL*, pages 118–126.

Markus Saers and Dekai Wu. 2009. Improving phrase-based translation via word alignments from stochastic inversion transduction grammars. In *Proceedings of the HLT-NAACL Workshop on Syntax and Structure in Statistical Translation*, pages 28–36.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-HLT*, pages 577–585.

Roy Tromble, Shankar Kumar, Franz J. Och, and Wolfgang Macherey. 2008. Lattice Minimum Bayes-Risk decoding for statistical machine translation. In *Proceedings of EMNLP*, pages 620–629.

Ashish Venugopal, Stephan Vogel, and Alex Waibel. 2003. Effective phrase translation extraction from alignment models. In *Proceedings of ACL*, pages 319–326.

Ashish Venugopal, Andreas Zollmann, Noah A. Smith, and Stephan Vogel. 2008. Wider pipelines: N-best alignments and parses in mt training. In *Proceedings of AMTA*, pages 192–201.

Andreas Zollmann, Ashish Venugopal, Franz J. Och, and Jay Ponte. 2008. A systematic comparison of phrase-based, hierarchical and syntax-augmented statistical MT. In *Proceedings of COLING*, pages 1145–1152.

Maximum Entropy Based Phrase Reordering for Hierarchical Phrase-based Translation

Zhongjun He Yao Meng Hao Yu
Fujitsu R&D Center CO., LTD.
15/F, Tower A, Ocean International Center, 56 Dongsihuan Zhong Rd.
Chaoyang District, Beijing, 100025, China
{hezhongjun, mengyao, yu}@cn.fujitsu.com

Abstract

Hierarchical phrase-based (HPB) translation provides a powerful mechanism to capture both short and long distance phrase reorderings. However, the phrase reorderings lack of contextual information in conventional HPB systems. This paper proposes a context-dependent phrase reordering approach that uses the maximum entropy (MaxEnt) model to help the HPB decoder select appropriate reordering patterns. We classify translation rules into several reordering patterns, and build a MaxEnt model for each pattern based on various contextual features. We integrate the MaxEnt models into the HPB model. Experimental results show that our approach achieves significant improvements over a standard HPB system on large-scale translation tasks. On Chinese-to-English translation, the absolute improvements in BLEU (case-insensitive) range from 1.2 to 2.1.

1 Introduction

The hierarchical phrase-based (HPB) model (Chiang, 2005; Chiang, 2007) has been widely adopted in statistical machine translation (SMT). It utilizes synchronous context free grammar (SCFG) rules to perform translation. Typically, there are three types of rules (see Table 1): *phrasal* rule, a phrase pair consisting of consecutive words; *hierarchical* rule, a hierarchical phrase pair consisting of both words and variables; and *glue* rule, which is used to merge phrases serially. Phrasal rule captures short distance reorderings within phrases, while hierarchical rule captures long distance reorderings between phrases. Therefore, the HPB model outperforms conventional phrase-based models on phrase reorderings.

Type	Constituent		Examples
	Word	Variable	
PR	√	-	$X \rightarrow \langle$之一, one of$\rangle$
HR	√	√	$X \rightarrow \langle X$的$, \text{of}X\rangle$
GR	-	√	$S \rightarrow \langle SX, SX\rangle$

Table 1: A classification of grammar rules for the HPB model. PR = phrasal rule, HR = hierarchical rule, GR = glue rule.

However, HPB translation suffers from a limitation, in that the phrase reorderings lack of contextual information, such as the surrounding words of a phrase and the content of sub-phrases that represented by variables. Consider the following two hierarchical rules in translating a Chinese sentence into English:

$$X \rightarrow \langle X_1 \text{ 的 } X_2,\ X_1 \text{ 's } X_2\rangle \quad (1)$$

$$X \rightarrow \langle X_1 \text{ 的 } X_2,\ X_2 X_1\rangle \quad (2)$$

和 俄罗斯 的 谈判
with Russia 's talks
talks with Russia

Both pattern-match the source sentence, but produce quite different phrase reorderings. The first rule generates a monotone translation, while the second rule swaps the source phrases covered by X_1 and X_2 on the target side. During decoding, the first

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 555–563,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

rule is more likely to be used, as it occurs more frequently in a training corpus. However, the example is not a noun possessive case because the subphrase covered by X_1 is not a noun but a prepositional phrase. Thus, without considering information of sub-phrases, the decoder may make errors on phrase reordering.

Contextual information has been widely used to improve translation performance. It is helpful to reduce ambiguity, thus guide the decoder to choose correct translation for a source text. Several researchers observed that word sense disambiguation improves translation quality on lexical translation (Carpuat and Wu, 2007; Chan et al., 2007). These methods utilized contextual features to determine the correct meaning of a source word, thus help an SMT system choose an appropriate target translation.

Zens and Ney (2006) and Xiong et al. (2006) utilized contextual information to improve phrase reordering. They addressed phrase reordering as a two-class classification problem that translating neighboring phrases serially or inversely. They built a maximum entropy (MaxEnt) classifier based on boundary words to predict the order of neighboring phrases.

He et al. (2008) presented a lexicalized rule selection model to improve both lexical translation and phrase reordering for HPB translation. They built a MaxEnt model for each ambiguous source side based on contextual features. The method was also successfully applied to improve syntax-based SMT translation (Liu et al., 2008), using more sophisticated syntactical features. Shen et al. (2008) integrated various contextual and linguistic features into an HPB system, using surrounding words and dependency information for building context and dependency language models, respectively.

In this paper, we focus on improving phrase reordering for HPB translation. We classify SCFG rules into several reordering patterns consisting of two variables X and F (or E) [1], such as X_1FX_2 and X_2EX_1. We treat phrase reordering as a classification problem and build a MaxEnt model for each source reordering pattern based on various contextual features. We propose a method to integrate the MaxEnt models into an HPB system. Specifically:

- For hierarchical rules, we classify the source-side and the target-side into 7 and 17 reordering patterns, respectively. Target reordering patterns are treated as possible labels. We then build a classifier for each source pattern to predict phrase reorderings. This is different from He et al. (2008), in which they built a classifier for each ambiguous hierarchical source-side. Therefore, the training examples for each MaxEnt model is small and the model maybe unstable. Here, we classify source hierarchical phrases into 7 reordering patterns according to the arrangement of words and variables. We can obtain sufficient samples for each MaxEnt model from large-scale bilingual corpus.

- For glue rules, we extend the HPB model by using bracketing transduction grammar (BTG) (Wu, 1996) instead of the monotone glue rule. By doing this, there are two options for the decoder to merge phrases: serial or inverse. We then build a classifier for glue rules to predict reorderings of neighboring phrases, analogous to Xiong et al. (2006).

- We integrate the MaxEnt based phrase reordering models as features into the HPB model (Chiang, 2005). The feature weights can be tuned together with other feature functions by MERT algorithm (Och, 2003).

Experimental results show that the presented method achieves significant improvement over the baseline. On Chinese-to-English translation tasks of NIST evluation, improvements in BLEU (case-insensitive) are 1.2 on MT06 GALE set, 1.8 on MT06 NIST set, and 2.1 on MT08.

The rest of the paper is structured as follows: Section 2 describes the MaxEnt based phrase reordering method. Section 3 integrates the MaxEnt models into the translation model. In Section 4, we report experimental results. We analyze the presented method and experimental results in Section 5 and conclude in Section 6.

[1] We use F and E to represent source and target words, respectively.

Source phrase	Target phrase
	X and
X 和	with X
	between X and

Figure 1: A source hierarchical phrase and its corresponding target translation.

2 MaxEnt based Phrase Reordering

We regard phrase reordering as a pattern classification problem. A reordering pattern indicates an arrangement of words and variables. Although there are a large amount of hierarchical rules may be extracted from bilingual corpus, these rules can be classified into several reordering patterns (Section 2.1). In addition, we extend the HPB model with BTG, that adding an inverted glue rule to merge phrases inversely (Section 2.2). Therefore, the glue rules are classified into two patterns: serial or inverse. We then build a MaxEnt phrase reordering (MEPR) classifier for each source reordering pattern (Section 2.3). In Section 2.4, we describe contextual features.

2.1 Reordering Pattern Classification for Hierarchical Rule

Hierarchical rule, consisting of both words and variables, is of great importance for the HPB model. During decoding, words are used for lexical translation, and variables capture phrase reordering. We may learn millions of hierarchical rules from a bilingual corpus. Although these rules are different from each other, they can be classified into several reordering patterns according to the arrangement of variables and words.

In this paper, we follow the constraint as described in (Chiang, 2005) that a hierarchical rule can have at most two variables and they cannot be adjacent on the source side. We use "X" to represent the variable, and "F" and "E" to represent word strings in source and target language, respectively. Therefore, in a hierarchical rule, E is the lexical translation of F, while the order of X and E contains phrase reordering information.

For the hierarchical rule that contains one variable (see Figure 1 for example), both the source and the target phrases can be classified into three patterns (Table 2). To reduce the complexity of classification, we do not distinguish the order of word strings. For example, we consider "e_1Xe_2" and "e_2Xe_1" as the same pattern "EXE", because the target words are determined by lexical translation of source words. Our focus is the order between X and E. During decoding the phrases covered by X are dynamically changed and the contextual information of these phrases is ignored for pattern-matching of hierarchical rules.

Source pattern	Target pattern
XF	XE
FX	EX
FXF	EXE

Table 2: A classification of the source side and the target side for the hierarchical rule that contains one variable.

Source pattern	Target pattern
	X_1EX_2
	X_2EX_1
	X_1X_2E
	X_2X_1E
	EX_1X_2
X_1FX_2	EX_2X_1
X_1FX_2F	X_1EX_2E
FX_1FX_2	X_2EX_1E
FX_1FX_2F	EX_1X_2E
	EX_2X_1E
	EX_1EX_2
	EX_2EX_1
	EX_1EX_2E
	EX_2EX_1E

Table 3: A classification of the source side and the target side for the hierarchical rule that contains two variables.

Analogously, for the hierarchical rule that contains two variables, the source phrases are classified into 4 patterns, while the target phrases are classified into 14 patterns, as shown in Table 3. The pattern number on the source side is less than that on the target side, because on the source side, "X_1" always appears before "X_2", and they cannot be adjacent.

2.2 Reordering Pattern Classification for Glue Rule

The HPB model used glue rule to combine phrases serially. The reason is that in some cases, there are no valid translation rules that cover a source span. Therefore, the glue rule provides a default monotone combination of phrases in order to complete a translation. This is not sufficient because in certain cases, the order of phrases may be inverted on the target-side.

In this paper, we extend the glue rule with BTG (Wu, 1996), which consists of three types of rules:

$$X \rightarrow \langle \tilde{f}, \tilde{e} \rangle \tag{3}$$

$$X \rightarrow \langle X_1 X_2, X_1 X_2 \rangle \tag{4}$$

$$X \rightarrow \langle X_1 X_2, X_2 X_1 \rangle \tag{5}$$

Rule 3 is a phrasal rule that translates a source phrase $\tilde{f}$ into a target phrase $\tilde{e}$. Rule 4 merges two consecutive phrases in monotone order, while Rule 5 merges them in inverted order. During decoding, the decoder first uses Rule 3 to produce phrase translation, and then iteratively uses Rule 4 and 5 to merge two neighboring phrases into a larger phrase until the whole sentence is covered.

We replace the original glue rules in the HPB model with BTG rules (see Table 4). We believe that the extended HPB model can benefit from BTG in the following aspects:

- In the HPB model, as we mentioned, hierarchical rules are constrained in that nonterminals cannot be adjacent on the source side, i.e., the source side cannot contain "$X_1 X_2$". One reason is that it will heavily increase the rule table size. The other reason is that it can cause a spurious ambiguity problem (Chiang, 2005). The inverted glue rule in BTG, however, can solve this problem.

- In the HPB model, only a monotone glue rule is provided to merge phrases serially. In the extended HPB model, the combination of phrases is classified into two types: monotone and inverse.

Analogous to Xiong et al. (2006), to perform context-dependent phrase reordering, we build a MaxEnt based classifier for glue rules to predict the order of two neighboring phrases. In this paper, we utilize more contextual features.

Glue Rule	Extended Glue Rule
$S \rightarrow \langle X, X \rangle$	$S \rightarrow \langle X, X \rangle$
$S \rightarrow \langle SX, SX \rangle$	$X \rightarrow \langle X_1 X_2, X_1 X_2 \rangle$
-	$X \rightarrow \langle X_1 X_2, X_2 X_1 \rangle$

Table 4: Extending the glue rules in the HPB model with BTG.

2.3 The MaxEnt based Phrase Reordering Classifier

As described above, we classified phrase reorderings into several patterns. Therefore, phrase reordering can be regarded as a classification problem: for each source reordering pattern, we treat the corresponding target reordering patterns as labels.

We build a general classification model within the MaxEnt framework:

$$P_{me}(T_\gamma | T_\alpha, \alpha, \gamma) = \frac{exp(\sum_i \lambda_i h_i(\gamma, \alpha, f(X), e(X))}{\sum_{T_\gamma} exp(\sum_i \lambda_i h_i(\gamma, \alpha, f(X), e(X))} \tag{6}$$

where, α and γ are the source and target side, respectively. T_α/T_γ is the reordering pattern of α/γ. $f(X)$ and $e(X)$ are the phrases that covered by X one the source and target side, respectively. Given a source phrase, the model predicts a target reordering pattern, considering various contextual features (Section 2.4).

According to the classification of reordering patterns, there are 3 kinds of classifiers:

- P_{me}^{hr1} includes 3 classifiers for the hierarchical rules that contain 1 variable. Each of the classifier has 3 labels;

- P_{me}^{hr2} includes 4 classifiers for the hierarchical rules that contain 2 variables. Each of the classifier has 14 labels;

- P_{me}^{gr} includes 1 classifier for the glue rules. The classifier has 2 labels that predict a monotone or inverse order for two neighboring phrases. This classifier is analogous to (Xiong et al., 2006).

There are 8 classifiers in total. This is much fewer than the classifiers in He et al. (2008), in which a classifier was built for each ambiguous hierarchical source side. In this way, a classifier may face the risk that there are not enough samples for training a stable MaxEnt model. While our approach is more generic, rather than training a MaxEnt model for a specific hierarchical source side, we train a model for a source reordering pattern. Thus, we reduce the number of classifiers and can extract large training examples for each classifier.

2.4 Feature definition

For a reordering pattern pair $\langle T_\alpha, T_\gamma \rangle$, we design three feature functions for phrase reordering classifiers:

- Source lexical feature, including boundary words and neighboring words. Boundary words are the left and right word of the source phrases covered by $f(X)$, while neighboring words are the words that immediately to the left and right of a source phrase $f(\alpha)$;
- Part-of-Speech (POS) feature, POS tags of the boundary and neighboring words on the source side.
- Target lexical feature, the boundary words of the target phrases covered by $e(X)$.

These features can be extracted together with translation rules from bilingual corpus. However, since the hierarchical rule does not allow for adjacent variables on the source side, we extract features for P_{me}^{gr} by using the method described in Xiong et al. (2006). We train the classifiers with a MaxEnt trainer (Zhang, 2004).

3 Integrating the MEPR Classifier into the HPB Model

The HPB model is built within the standard log-linear framework (Och and Ney, 2002):

$$Pr(e|f) \propto \sum_i \lambda_i h_i(\alpha, \gamma) \tag{7}$$

where $h_i(\alpha, \gamma)$ is a feature function and λ_i is the weight of h_i. The HPB model has the following features: translation probabilities $p(\gamma|\alpha)$ and $p(\alpha|\gamma)$, lexical weights $p_w(\gamma|\alpha)$ and $p_w(\alpha|\gamma)$, word penalty, phrase penalty, glue rule penalty, and a target n-gram language model.

To integrate the MEPR classifiers into the translation model, the features of the log-linear model are changed as follows:

- We add the MEPR classifier as a feature function to predict reordering pattern:

  $$h_{me}(T_\gamma|T_\alpha) = \sum P_{me}(T_\gamma|T_\alpha, \alpha, \gamma) \tag{8}$$

  During decoding, we first classify each source phrase into one of the 8 source reordering patterns and then use the corresponding MEPR classifier to predict the possible target reordering pattern. Therefore, the contextual information guides the decoder to perform phrase reordering.

- We split the "glue rule penalty" into two features: monotone glue rule number and inverted glue rule number. These features reflect preference of the decoder for using monotone or inverted glue rules.

The advantage of our extension method is that the weights of the new features can be tuned together with the other features by MERT algorithm (Och, 2003).

We utilize a standard CKY algorithm for decoding. Given a source sentence, the decoder searches the best derivation from the bottom to top. For a source span $[j_1, j_2]$, the decoder uses three kinds of rules: translation rules produce lexical translation and phrase reordering (for hierarchical rules), monotone rule merges any neighboring sub-spans $[j_1, k]$ and $[k+1, j_2]$ serially, and inverted rule swap them. Note that when the decoder uses the monotone and inverted glue rule to combine sub-spans, it merges phrases that do not contain variables. Because the CKY algorithm guarantees that the sub spans $[j_1, k]$ and $[k+1, j_2]$ have been translated before $[j_1, j_2]$.

4 Experiments

We carried out experiments on four systems:

- HPB: replication of the Hiero system (Chiang, 2005);
- HPB+MEHR: HPB with MaxEnt based classifier for hierarchical rules, as described in Section 2.1;
- HPB+MEGR: HPB with MaxEnt based classifier for glue rules, as described in Section 2.2;
- HPB+MER: HPB with MaxEnt based classifier for both hierarchical and glue rules.

All systems were tuned on NIST MT03 and tested on MT06 and MT08. The evaluation metric was BLEU (Papineni et al., 2002) with case-insensitive matching of n-grams, where $n = 4$.

We evaluated our approach on Chinese-to-English translation. The training data contained 77M Chinese words and 81M English words. These data come from 17 corpora: LDC2002E18, LDC2002L27, LDC2002T01, LDC2003E07, LDC2003E14, LDC2004T07, LDC2005E83, LDC2005T06, LDC2005T10, LDC2005T34, LDC2006E24, LDC2006E26, LDC2006E34, LDC2006E86, LDC2006E92, LDC2006E93, LDC2004T08 (HK_News, HK_Hansards).

To obtain word alignments, we first ran GIZA++ (Och and Ney, 2000) in both translation directions and then refined the results using the "grow-diag-final" method (Koehn et al., 2003). For the language model, we used the SRI Language Modeling Toolkit (Stolcke, 2002) to train two 4-gram models on the Xinhua portion of the GigaWord corpus and the English side of the training corpus.

4.1 Statistical Information of Rules

Hierarchical Rules

We extracted 162M translation rules from the training corpus. Among them, there were 127M hierarchical rules, which contained 85M hierarchical source phrases. We classified these source phrases into 7 patterns as described in Section 2.1. Table 5 shows the statistical information. We observed that the most frequent source pattern is "FXF", which accounted for 46.1% of the total. Interestingly, "X_1FX_2", accounting for 3.7%, was the least frequent pattern. Table 6 and Table 7 show the distributions of reordering patterns for hierarchical source phrases that contain one and two variables, respectively. From both the tables, we observed that for Chinese-to-English translation, the most frequent "reordering" pattern for a source phrase is monotone translation (bold font in the tables).

Source Pattern	Percentage (%)
XF	9.7
FX	9.7
FXF	46.1
X_1FX_2	3.7
X_1FX_2F	11.9
FX_1FX_2	11.8
FX_1FX_2F	7.1

Table 5: Statistical information of reordering pattern classification for hierarchical source phrases.

#	Source		
Target (%)	FX	XF	FXF
EX	**82.8**	7	4.6
XE	6.4	**82.4**	2.9
EXE	10.8	10.6	**92.5**

Table 6: Percentage of target reordering pattern for each source pattern containing one variable.

Glue Rules

To train a MaxEnt classifier for glue rules, we extracted 65.8M reordering (monotone and inverse) instances from the training data, using the algorithm described in Xiong et al. (2006). There were 63M monotone instances, accounting for 95.7%. Although instances of inverse reordering accounted for 4.3%, they are important for phrase reordering.

4.2 Results

Table 8 shows the BLEU scores and decoding speed of the four systems on MT06 (GALE set and NIST set) and MT08. From the table, we made the following observations:

#	Source			
Target (%)	FX_1FX_2	FX_1FX_2F	X_1FX_2	X_1FX_2F
EX_1EX_2	**78.1**	3.6	4.6	1.2
EX_1EX_2E	2.1	**75.9**	0.1	1.6
EX_1X_2	6.8	0.1	2.8	0.1
EX_1X_2E	1.8	11.2	0.1	2
EX_2EX_1	2.8	1.4	2	1.2
EX_2EX_1E	1.4	2.3	0.7	1.1
EX_2X_1	0.9	0.1	2.2	0.2
EX_2X_1E	1	1.1	0.9	1.0
X_1EX_2	1.9	0.1	**71.2**	3.3
X_1EX_2E	0.7	2.1	6	**78.4**
X_1X_2E	0.1	0.1	2.8	5.9
X_2EX_1	0.9	0.4	1.6	0.7
X_2EX_1E	1.5	1.5	2.6	2.4
X_2X_1E	0.1	0.04	2.2	0.8

Table 7: Percentage of target reordering pattern for each source pattern containing two variables.

System	Test Data			Speed
	06G	06N	08	
HPB	14.19	33.93	25.85	8.7
HPB+MEHR	14.76	34.95	26.56	3.2
HPB+MEGR	15.09	35.72	27.34	2.7
HPB+MER	15.42	35.80	27.94	1.7

Table 8: BLEU percentage scores and translation speed (words/second) on test data. G=GALE set, N=NIST set. All improvements are statistically significant ($p < 0.01$). Note that MT06G has one reference for each source sentence, while the MT06N and MT08 have four references.

- The HPB+MEHR system achieved significant improvements on all test sets compared to the HPB system. The absolute increases in BLEU scores ranging from 0.6 (on 06G) to 1.0 (on 06N) percentage points. This indicates that the ME based reordering for hierarchical rules improves translation performance.

- The HPB+MEGR system achieved significant improvements over the HPB system. The absolute increases in BLEU scores ranging from 0.9 (on 06G) to 1.8 (on 06N) percentage points. The HPB+MEGR system overcomes the shortcoming of the HPB system by using both monotone glue rule and inverted glue rule, which merging phrases serially and inversely, respectively. Furthermore, the HPB+MEGR system outperformed the HPB+MEHR system.

- The HPB+MER system achieved the best performances on all test sets, with absolute increases of BLEU scores ranging from 1.2 (on 06G) to 2.1 (on 08). The system combining with ME based reordering for both hierarchical and glue rules, outperformed both the HPB+MEHR and HPB+MEGR systems.

- In addition, we found that the decoder takes more time after adding the MEPR models (the *speed* column of Table 8). The average translation speed of HPB+MER (1.7 words/second) is about 5 times slower than the HPB system (8.7 words/second). One reason is that the MEPR models utilized contextual information to compute classification scores. Another reason is that adding inverted glue rules increases search space.

5 Analysis

Experiments showed that the presented approach achieved significant gains on BLEU scores. Furthermore, we sought to explore what would happen after integrating the MEPR classifiers into the translation model. We compared the outputs of HPB and HPB+MER and observed that the translation performance are improved on phrase reordering. For example, the translations of a source sentence in MT08 are as follows [2]:

- **Src:** 韩国$_1$ 政府$_2$ 上个月$_3$ 底$_4$ 开始$_5$ 启动$_6$ 对$_7$ 朝鲜$_8$ 提供$_9$ 40万$_{10}$ 吨$_{11}$ 大米$_{12}$ 的$_{13}$ 援助$_{14}$ 计划$_{15}$

- **Ref:** At the end$_4$ of last$_3$ month$_3$, the South$_1$ Korean$_1$ government$_2$ began$_5$ a plan$_{15}$ to provide$_9$ 400,000$_{10}$ tonnes$_{11}$ of rice$_{12}$ as aid$_{14}$ to North$_8$ Korea$_8$

- **HPB:** South Korean government late last month to start with 400,000 tons of rice aid to the DPRK

- **HPB+MER**: Start at the end of last month, South Korean government plans to provide 400,000 tons of rice in aid to the DPRK

The most obvious error that the baseline system makes is the order of the time expression "*上个月底, the end of last month*", which should be either at the beginning or the end on target side. However, the baseline produced a monotone translation by using the rule "*韩国 政府* X_1, *South Korean government* X_1". The HPB+MER system, however, moved the time expression to the beginning of the sentence by using the rule "*韩国 政府* X_1, X_1 *South Korean government*'. The reason is that the MaxEnt phrase reordering classifier uses the contextual features (e.g. the boundary words) of the phrase covered by X_1 to predict the phrase reordering as X_1E for the source phrase FX_1.

6 Conclusions and Future Work

In this paper, we have proposed a MaxEnt based phrase reordering approach to help the HPB decoder select reordering patterns. We classified hierarchical rules into 7 reordering patterns on the source side and 17 reordering patterns on the target side. In addition, we introduced BTG to enhance the reordering of neighboring phrases and classified the glue rules into two patterns. We trained a MaxEnt classifier for each reordering pattern and integrated it into a standard HPB system. Experimental results showed that the proposed approach achieved significant improvements over the baseline. The absolute improvements in BLEU range from 1.2 to 2.1.

MaxEnt based phrase reordering provides a mechanism to incorporate various features into the translation model. In this paper, we only use a few feature sets based on standard contextual word and POS tags. We believe that additional features will further improve translation performance. Such features could include syntactical features (Chiang et al., 2009). In the future, we will carry out experiments on deeper features and evaluate the effects of different feature sets.

References

Marine Carpuat and Dekai Wu. 2007. Improving statistical machine translation using word sense disambiguation. In *Proceedings of EMNLP-CoNLL 2007*, pages 61–72.

Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007. Word sense disambiguation improves statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 33–40.

David Chiang, Wei Wang, and Kevin Knight. 2009. 11,001 new features for statistical machine translation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies 2009 Conference*, page 218 – 226.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 263–270.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, pages 33(2):201–228.

Zhongjun He, Qun Liu, and Shouxun Lin. 2008. Improving statistical machine translation using lexical-

[2] The co-indexes of the words in the source and reference sentence indicate word alignments.

ized rule selection. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 321–328.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54.

Qun Liu, Zhongjun He, Yang Liu, and Shouxun Lin. 2008. Maximum entropy based rule selection model for syntax-based statistical machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, page 89﹣97.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 440–447.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Libin Shen, Jinxi Xu, Bing Zhang, Spyros Matsoukas, and Ralph Weischedel. 2008. Effective use of linguistic and contextual information for statistical machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 72–80.

Andreas Stolcke. 2002. SRILM – An extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken language Processing*, volume 2, pages 901–904.

Dekai Wu. 1996. A polynomial-time algorithm for statistical machine translation. In *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*, pages 521–528.

Richard Zens and Hermann Ney. 2006. Discriminative reordering models for statistical machine translation. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 55–63.

Le Zhang. 2004. Maximum entropy modeling toolkit for python and c++. available at http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html.

Further Meta-Evaluation of Broad-Coverage Surface Realization

Dominic Espinosa and **Rajakrishnan Rajkumar** and **Michael White** and **Shoshana Berleant**
Department of Linguistics
The Ohio State University
Columbus, Ohio, USA
{espinosa,raja,mwhite,berleant}@ling.ohio-state.edu

Abstract

We present the first evaluation of the utility of automatic evaluation metrics on surface realizations of Penn Treebank data. Using outputs of the OpenCCG and XLE realizers, along with ranked WordNet synonym substitutions, we collected a corpus of generated surface realizations. These outputs were then rated and post-edited by human annotators. We evaluated the realizations using seven automatic metrics, and analyzed correlations obtained between the human judgments and the automatic scores. In contrast to previous NLG meta-evaluations, we find that several of the metrics correlate moderately well with human judgments of both adequacy and fluency, with the TER family performing best overall. We also find that all of the metrics correctly predict more than half of the significant system-level differences, though none are correct in all cases. We conclude with a discussion of the implications for the utility of such metrics in evaluating generation in the presence of variation. A further result of our research is a corpus of post-edited realizations, which will be made available to the research community.

1 Introduction and Background

In building surface-realization systems for natural language generation, there is a need for reliable automated metrics to evaluate the output. Unlike in parsing, where there is usually a single gold-standard parse for a sentence, in surface realization there are usually many grammatically-acceptable ways to express the same concept. This parallels the task of evaluating machine-translation (MT) systems: for a given segment in the source language, there are usually several acceptable translations into the target language. As human evaluation of translation quality is time-consuming and expensive, a number of automated metrics have been developed to evaluate the quality of MT outputs. In this study, we investigate whether the metrics developed for MT evaluation tasks can be used to reliably evaluate the outputs of surface realizers, and which of these metrics are best suited to this task.

A number of surface realizers have been developed using the Penn Treebank (PTB), and BLEU scores are often reported in the evaluations of these systems. But how useful is BLEU in this context? The original BLEU study (Papineni et al., 2001) scored MT outputs, which are of generally lower quality than grammar-based surface realizations. Furthermore, even for MT systems, the usefulness of BLEU has been called into question (Callison-Burch et al., 2006). BLEU is designed to work with multiple reference sentences, but in treebank realization, there is only a single reference sentence available for comparison.

A few other studies have investigated the use of such metrics in evaluating the output of NLG systems, notably (Reiter and Belz, 2009) and (Stent et al., 2005). The former examined the performance of BLEU and ROUGE with computer-generated weather reports, finding a moderate correlation with human fluency judgments. The latter study applied several MT metrics to paraphrase data from Barzilay and Lee's corpus-based system (Barzilay and Lee, 2003), and found moderate correlations with human adequacy judgments, but little correlation with fluency judgments. Cahill (2009) examined the performance of six MT metrics (including BLEU) in evaluating the output of a LFG-based surface realizer for

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 564–574,
MIT, Massachusetts, USA, 9-11 October 2010.

German, also finding only weak correlations with the human judgments.

To study the usefulness of evaluation metrics such as BLEU on the output of grammar-based surface realizers used with the PTB, we assembled a corpus of surface realizations from three different realizers operating on Section 00 of the PTB. Two human judges evaluated the adequacy and fluency of each of the realizations with respect to the reference sentence. The realizations were then scored with a number of automated evaluation metrics developed for machine translation. In order to investigate the correlation of targeted metrics with human evaluations, and gather other acceptable realizations for future evaluations, the judges manually repaired each unacceptable realization during the rating task. In contrast to previous NLG meta-evaluations, we found that several of the metrics correlate moderately well with human judgments of both adequacy and fluency, with the TER family performing best. However, when looking at statistically significant system-level differences in human judgments, we found that some of the metrics get some of the rankings correct, but none get them all correct, with different metrics making different ranking errors. This suggests that multiple metrics should be routinely consulted when comparing realizer systems.

Overall, our methodology is similar to that of previous MT meta-evaluations, in that we collected human judgments of system outputs, and compared these scores with those assigned by automatic metrics. A recent alternative approach to paraphrase evaluation is ParaMetric (Callison-Burch et al., 2008); however, it requires a corpus of annotated (aligned) paraphrases (which does not yet exist for PTB data), and is arguably focused more on paraphrase analysis than paraphrase generation.

The plan of the paper is as follows: Section 2 discusses the preparation of the corpus of surface realizations. Section 3 describes the human evaluation task and the automated metrics applied. Sections 4 and 5 present and discuss the results of these evaluations. We conclude with some general observations about automatic evaluation of surface realizers, and some directions for further research.

2 Data Preparation

We collected realizations of the sentences in Section 00 of the WSJ corpus from the following three sources:

1. OpenCCG, a CCG-based chart realizer (White, 2006)
2. The XLE Generator, a LFG-based system developed by Xerox PARC (Crouch et al., 2008)
3. WordNet synonym substitutions, to investigate how differences in lexical choice compare to grammar-based variation.[1]

Although all three systems used Section 00 of the PTB, they were applied with various parameters (e.g., language models, multiple-output versus single-output) and on different input structures. Accordingly, our study does not compare OpenCCG to XLE, or either of these to the WordNet system.

2.1 OpenCCG realizations

OpenCCG is an open source parsing/realization library with multimodal extensions to CCG (Baldridge, 2002). The OpenCCG chart realizer takes logical forms as input and produces strings by combining signs for lexical items. Alternative realizations are scored using integrated n-gram and perceptron models. For robustness, fragments are greedily assembled when necessary. Realizations were generated from 1,895 gold standard logical forms, created by constrained parsing of development-section derivations. The following OpenCCG models (which differ essentially in the way the output is ranked) were used:

1. Baseline 1: Output ranked by a trigram word model
2. Baseline 2: Output ranked using three language models (3-gram words + 3-gram words with named entity class replacement + factored language model of words, POS tags and CCG supertags)

[1] Not strictly surface realizations, since they do not involve an abstract input specification, but for simplicity we refer to them as realizations throughout.

3. Baseline 3: Perceptron with syntax features and the three LMs mentioned above

4. Perceptron full-model: n-best realizations ranked using perceptron with syntax features and the three n-gram models, as well as discriminative n-grams

The perceptron model was trained on sections 02-21 of the CCGbank, while a grammar extracted from section 00-21 was used for realization. In addition, oracle supertags were inserted into the chart during realization. The purpose of such a non-blind testing strategy was to evaluate the quality of the output produced by the statistical ranking models in isolation, rather than focusing on grammar coverage, and avoid the problems associated with lexical smoothing, i.e. lexical categories in the development section not being present in the training section.

To enrich the variation in the generated realizations, dative-alternation was enforced during realization by ensuring alternate lexical categories of the verb in question, as in the following example:

(1) the executives gave [the chefs] [a standing ovation]

(2) the executives gave [a standing ovation] [to the chefs]

2.2 XLE realizations

The corpus of realizations generated by the XLE system contained 42,527 surface realizations of approximately 1,421 section 00 sentences (an average of 30 per sentence), initially unranked. The LFG f-structures used as input to the XLE generator were derived from automatic parses, as described in (Riezler et al., 2002). The realizations were first tokenized using Penn Treebank conventions, then ranked using perplexities calculated from the same trigram word model used with OpenCCG. For each sentence, the top 4 realizations were selected. The XLE generator provides an interesting point of comparison to OpenCCG as it uses a manually-developed grammar with inputs that are less abstract but potentially noisier, as they are derived from automatic parses rather than gold-standard ones.

2.3 WordNet synonymizer

To produce an additional source of variation, the nouns and verbs of the sentences in section 00 of the PTB were replaced with all of their WordNet synonyms. Verb forms were generated using verb stems, part-of-speech tags, and the *morphg* tool.[2] These substituted outputs were then filtered using the n-gram data which Google Inc. has made available.[3] Those without any 5-gram matches centered on the substituted word (or 3-gram matches, in the case of short sentences) were eliminated.

3 Evaluation

From the data sources described in the previous section, a corpus of realizations to be evaluated by the human judges was constructed by randomly choosing 305 sentences from section 00, then selecting surface realizations of these sentences using the following algorithm:

1. Add OpenCCG's best-scored realization.

2. Add other OpenCCG realizations until all four models are represented, to a maximum of 4.

3. Add up to 4 realizations from either the XLE system or the WordNet pool, chosen randomly.

The intent was to give reasonable coverage of all realizer systems discussed in Section 2 without overloading the human judges. "System" here means any instantiation that emits surface realizations, including various configurations of OpenCCG (using different language models or ranking systems), and these can be multiple-output, such as an n-best list, or single-output (best-only, worst-only, etc.). Accordingly, more realizations were selected from the OpenCCG realizer because 5 different systems were being represented. Realizations were chosen randomly, rather than according to sentence types or other criteria, in order to produce a representative sample of the corpus. In total, 2,114 realizations were selected for evaluation.

[2] http://www.informatics.sussex.ac.uk/research/groups/nlp/carroll/morph.html

[3] http://www.ldc.upenn.edu/Catalog/docs/LDC2006T13/readme.txt

3.1 Human judgments

Two human judges evaluated each surface realization on two criteria: *adequacy*, which represents the extent to which the output conveys all and only the meaning of the reference sentence; and *fluency*, the extent to which it is grammatically acceptable. The realizations were presented to the judges in sets containing a reference sentence and the 1-8 outputs selected for that sentence. To aid in the evaluation of adequacy, one sentence each of leading and trailing context were displayed. Judges used the guidelines given in Figure 1, based on the scales developed by the NIST Machine Translation Evaluation Workshop.

In addition to rating each realization on the two five-point scales, each judge also repaired each output which he or she did not judge to be fully adequate and fluent. An example is shown in Figure 2. These repairs resulted in new reference sentences for a substantial number of sentences. These repaired realizations were later used to calculate *targeted* versions of the evaluation metrics, i.e., using the repaired sentence as the reference sentence. Although targeted metrics are not fully automatic, they are of interest because they allow the evaluation algorithm to focus on what is actually wrong with the input, rather than all textual differences. Notably, targeted TER (HTER) has been shown to be more consistent with human judgments than human annotators are with one another (Snover et al., 2006).

3.2 Automatic evaluation

The realizations were also evaluated using seven automatic metrics:

- IBM's BLEU, which scores a hypothesis by counting n-gram matches with the reference sentence (Papineni et al., 2001), with smoothing as described in (Lin and Och, 2004)
- The NIST n-gram evaluation metric, similar to BLEU, but rewarding rarer n-gram matches, and using a different length penalty
- METEOR, which measures the harmonic mean of unigram precision and recall, with a higher weight for recall (Banerjee and Lavie, 2005)
- TER (Translation Edit Rate), a measure of the number of edits required to transform a hypothesis sentence into the reference sentence (Snover et al., 2006)
- TERP, an augmented version of TER which performs phrasal substitutions, stemming, and checks for synonyms, among other improvements (Snover et al., 2009)
- TERPA, an instantiation of TERP with edit weights optimized for correlation with adequacy in MT evaluations
- GTM (General Text Matcher), a generalization of the F-measure that rewards contiguous matching spans (Turian et al., 2003)

Additionally, targeted versions of BLEU, METEOR, TER, and GTM were computed by using the human-repaired outputs as the reference set. The human repair was different from the reference sentence in 193 cases (about 9% of the total), and we expected this to result in better scores and correlations with the human judgments overall.

4 Results

4.1 Human judgments

Table 1 summarizes the dataset, as well as the mean adequacy and fluency scores garnered from the human evaluation. Overall adequacy and fluency judgments were high (4.16, 3.63) for the realizer systems on average, and the best-rated realizer systems achieved mean fluency scores above 4.

4.2 Inter-annotator agreement

Inter-annotator agreement was measured using the κ-coefficient, which is commonly used to measure the extent to which annotators agree in category judgment tasks. κ is defined as $\frac{P(A)-P(E)}{1-P(E)}$, where $P(A)$ is the observed agreement between annotators and $P(E)$ is the probability of agreement due to chance (Carletta, 1996). Chance agreement for this data is calculated by the method discussed in Carletta's squib. However, in previous work in MT meta-evaluation, Callison-Burch et al. (2007), assume the less strict criterion of uniform chance agreement, i.e. $\frac{1}{5}$ for a five-point scale. They also

Score	*Adequacy*	*Fluency*
5	All the meaning of the reference	Perfectly grammatical
4	Most of the meaning	Awkward or non-native; punctuation errors
3	Much of the meaning	Agreement errors or minor syntactic problems
2	Meaning substantially different	Major syntactic problems, such as missing words
1	Meaning completely different	Completely ungrammatical

Figure 1: Rating scale and guidelines

Ref.	It wasn't clear how NL and Mr. Simmons would respond if Georgia Gulf spurns them again
Realiz.	It weren't clear how NL and Mr. Simmons would respond if Georgia Gulf again spurns them
Repair	It wasn't clear how NL and Mr. Simmons would respond if Georgia Gulf again spurns them

Figure 2: Example of repair

introduce the notion of "relative" κ, which measures how often two or more judges agreed that $A > B$, $A = B$, or $A < B$ for two outputs A and B, irrespective of the specific values given on the five-point scale; here, uniform chance agreement is taken to be $\frac{1}{3}$. We report both absolute and relative κ in Table 2, using actual chance agreement rather than uniform chance agreement.

The κ scores of 0.60 for adequacy and 0.63 for fluency across the entire dataset represent "substantial" agreement, according to the guidelines discussed in (Landis and Koch, 1977), better than is typically reported for machine translation evaluation tasks; for example, Callison-Burch et al. (2007) reported "fair" agreement, with κ = 0.281 for fluency and κ = 0.307 for adequacy (relative). Assuming the uniform chance agreement that the previously cited work adopts, our inter-annotator agreements (both absolute and relative) are still higher. This is likely due to the generally high quality of the realizations evaluated, leading to easier judgments.

4.3 Correlation with automatic evaluation

To determine how well the automatic evaluation methods described in Section 3 correlate with the human judgments, we averaged the human judgments for adequacy and fluency, respectively, for each of the rated realizations, and then computed both Pearson's correlation coefficient and Spearman's rank correlation coefficient between these scores and each of the metrics. Spearman's correlation makes fewer assumptions about the distribution of the data, but may not reflect a linear relationship that is actually present. Both are frequently reported in the literature. Due to space constraints, we show only Spearman's correlation, although the TER family scored slightly better on Pearson's coefficient, relatively.

The results for Spearman's correlation are given in Table 3. Additionally, the average scores for adequacy and fluency were themselves averaged into a single score, following (Snover et al., 2009), and the Spearman's correlation of each of the automatic metrics with these scores are given in Table 4. All reported correlations are significant at $p < 0.001$.

4.4 Bootstrap sampling of correlations

For each of the sub-corpora shown in Table 1, we computed confidence intervals for the correlations between adequacy and fluency human scores with selected automatic metrics (BLEU, HBLEU, TER, TERP, and HTER) as described in (Koenh, 2004). We sampled each sub-corpus 1000 times with replacement, and calculated correlations between the rankings induced by the human scores and those induced by the metrics for each reference sentence. We then used these coefficients to estimate the confidence interval, after excluding the top 25 and bottom 25 coefficients, following (Lin and Och, 2004). The results of this for the BLEU metric are shown in Table 5. We determined which correlations lay within the 95% confidence interval of the best performing metric in each row of Table Table 3; these figures are italicized.

5 Discussion

5.1 Human judgments of systems

The results for the four OpenCCG perceptron models mostly confirm those reported in (White and Rajkumar, 2009), with one exception: the B-3 model was below B-2, though the P-B (perceptron-best) model still scored highest. This may have been due to differences in the testing scenario. None of the differences in adequacy scores among the individual systems are significant, with the exception of the WordNet system. In this case, the lack of word-sense disambiguation for the substituted words results in a poor overall adequacy score (e.g., *wage floor* → *wage story*). Conversely, it scores highest for fluency, as substituting a noun or verb with a synonym does not usually introduce ungrammaticality.

5.2 Correlations of human judgments with MT metrics

Of the non-human-targeted metrics evaluated, BLEU and TER/TERP demonstrate the highest correlations with the human judgments of fluency ($r = 0.62, 0.64$). The TER family of evaluation metrics have been observed to perform very well in MT-evaluation tasks, and although the data evaluated here differs from typical MT data in some important ways, the correlation of TERP with the human judgments is substantial. In contrast with previous MT evaluations where TERP performs considerably better than TER, these scored close to equal on our data, possibly because TERP's stem, synonym, and paraphrase matching are less useful when most of the variation is syntactic.

The correlations with BLEU and METEOR are lower than those reported in (Callison-Burch et al., 2007); in that study, BLEU achieved adequacy and fluency correlations of 0.690 and 0.722, respectively, and METEOR achieved 0.701 and 0.719. The correlations for these metrics might be expected to be lower for our data, since overall quality is higher, making the metrics' task more difficult as the outputs involve subtler differences between acceptable and unacceptable variation.

The human-targeted metrics (represented by the prefixed *H* in the data tables) correlated even more strongly with the human judgments, compared to the non-targeted versions. HTER demonstrated the best correlation with realizer fluency ($r = 0.75$).

For several kinds of acceptable variation involving the rearrangement of constituents (such as dative shift), TERP gives a more reasonable score than BLEU, due to its ability to directly evaluate phrasal shifts. The following realization was rated 4.5 for fluency, and was more correctly ranked by TERP than BLEU:

(3) *Ref:* The deal also gave Mitsui access to a high-tech medical product.

(4) *Realiz.:* The deal also gave access to a high-tech medical product to Mitsui.

For each reference sentence, we compared the ranking of its realizations induced from the human scores to the ranking induced from the TERP score, and counted the rank errors by the latter, informally categorizing them by error type (see Table 7). In the 50 sentences with the highest numbers of rank errors, 17 were affected by punctuation differences, typically involving variation in comma placement. Human fluency judgments of outputs with only punctuation problems were generally high, and many realizations with commas inserted or removed were rated fully fluent by the annotators. However, TERP penalizes such insertions or deletions. Agreement errors are another frequent source of ranking errors for TERP. The human judges tended to harshly penalize sentences with number-agreement or tense errors, whereas TERP applies only a single substitution penalty for each such error. We expect that with suitable optimization of edit weights to avoid over-penalizing punctuation shifts and under-penalizing agreement errors, TERP would exhibit an even stronger correlation with human fluency judgments.

None of the evaluation metrics can distinguish an acceptable movement of a word or constituent from an unacceptable movement, with only one reference sentence. A substantial source of error for both TERP and BLEU is variation in adverbial placement, as shown in (7).

Similar errors are seen with prepositional phrases and some commonly-occurring temporal adverbs, which typically admit a number of variations in placement. Another important example of acceptable variation which these metrics do not generally rank correctly is dative alternation:

(7) *Ref.* We need to clarify what exactly is wrong with it.

Realiz.	*Flu.*	TERP	BLEU
We need to clarify exactly what is wrong with it.	5	0.1	0.5555
We need to clarify exactly what 's wrong with it.	5	0.2	0.4046
We need to clarify what , exactly , is wrong with it.	5	0.2	0.5452
We need to clarify what is wrong with it exactly.	4.5	0.1	0.6756
We need to clarify what exactly , is wrong with it.	4	0.1	0.7017
We need to clarify what , exactly is wrong with it.	4	0.1	0.7017
We needs to clarify exactly what is wrong with it.	3	0.103	0.346

(5) *Ref.* When test booklets were passed out 48 hours ahead of time, she says she copied questions in the social studies section and gave the answers to students.

(6) *Realiz.* When test booklets were passed out 48 hours ahead of time , she says she copied questions in the social studies section and gave students the answers.

The correlations of each of the metrics with the human judgments of fluency for the realizer systems indicate at least a moderate relationship, in contrast with the results reported in (Stent et al., 2005) for paraphrase data, which found an inverse correlation for fluency, and (Cahill, 2009) for the output of a surface realizer for German, which found only a weak correlation. However, the former study employed a corpus-based paraphrase generation system rather than grammar-driven surface realizers, and the resulting paraphrases exhibited much broader variation. In Cahill's study, the outputs of the realizer were almost always grammatically correct, and the automated evaluation metrics were ranking markedness instead of grammatical acceptability.

5.3 System-level comparisons

In order to investigate the efficacy of the metrics in ranking different realizer systems, or competing realizations from the same system generated using different ranking models, we considered seven different "systems" from the whole dataset of realizations. These consisted of five OpenCCG-based realizations (the best realization from three baseline models, and the best and the worst realization from the full perceptron model), and two XLE-based systems (the best and the worst realization, after ranking the outputs of the XLE realizer with an n-gram model). The mean of the combined adequacy and fluency scores of each of these seven systems was compared with that of every other system, resulting in 21 pairwise comparisons. Then Tukey's HSD test was performed to determine the systems which differed significantly in terms of the average adequacy and fluency rating they received.[4] The test revealed five pairwise comparisons where the scores were significantly different.

Subsequently, for each of these systems, an overall system-level score for each of the MT metrics was calculated. For the five pairwise comparisons where the adequacy-fluency group means differed significantly, we checked whether the metric ranked the systems correctly. Table 8 shows the results of a pairwise comparison between the ranking induced by each evaluation metric, and the ranking induced by the human judgments. Five of the seven non-targeted metrics correctly rank more than half of the systems. NIST, METEOR, and GTM get the most comparisons right, but neither NIST nor GTM correctly rank the OpenCCG-baseline model 1 with respect to the XLE-best model. TER and TERP get two of the five comparisons correct, and they incorrectly rank two of the five OpenCCG model comparisons, as well as the comparison between the XLE-worst and OpenCCG-best systems.

For the targeted metrics, HNIST is correct for all five comparisons, while neither HBLEU nor HMETEOR correctly rank all the OpenCCG models. On the other hand, HTER and HGTM incorrectly rank the XLE-best system versus OpenCCG-based models.

In summary, some of the metrics get some of the rankings correct, but none of the non-targeted metrics get all of them correct. Moreover, different metrics make different ranking errors. This argues for

[4] This particular test was chosen since it corrects for multiple post-hoc analyses conducted on the same data-set.

the use of multiple metrics in comparing realizer systems.

6 Conclusion

Our study suggests that although the task of evaluating the output from realizer systems differs from the task of evaluating machine translations, the automatic metrics used to evaluate MT outputs deliver moderate correlations with combined human fluency and adequacy scores when used on surface realizations. We also found that the MT-evaluation metrics are useful in evaluating different versions of the same realizer system (e.g., the various OpenCCG realization ranking models), and finding cases where a system is performing poorly. As in MT-evaluation tasks, human-targeted metrics have the highest correlations with human judgments overall. These results suggest that the MT-evaluation metrics are useful for developing surface realizers. However, the correlations are lower than those reported for MT data, suggesting that they should be used with caution, especially for cross-system evaluation, where consulting multiple metrics may yield more reliable comparisons. In our study, the targeted version of TERP correlated most strongly with human judgments of fluency.

In future work, the performance of the TER family of metrics on this data might be improved by optimizing the edit weights used in computing its scores, so as to avoid over-penalizing punctuation movements or under-penalizing agreement errors, both of which were significant sources of ranking errors. Multiple reference sentences may also help mitigate these problems, and the corpus of human-repaired realizations that has resulted from our study is a step in this direction, as it provides multiple references for some cases. We expect the corpus to also prove useful for feature engineering and error analysis in developing better realization models.[5]

Acknowledgements

We thank Aoife Cahill and Tracy King for providing us with the output of the XLE generator. We also thank Chris Callison-Burch and the anonymous reviewers for their helpful comments and suggestions.

This material is based upon work supported by the National Science Foundation under Grant No. 0812297.

References

Jason Baldridge. 2002. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

S. Banerjee and A. Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72.

R. Barzilay and L. Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *proceedings of HLT-NAACL*, volume 2003, pages 16–23.

Aoife Cahill. 2009. Correlating human and automatic evaluation of a german surface realiser. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 97–100, Suntec, Singapore, August. Association for Computational Linguistics.

C. Callison-Burch, M. Osborne, and P. Koehn. 2006. Reevaluating the role of BLEU in machine translation research. In *Proceedings of EACL*, volume 2006, pages 249–256.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2007. (meta-) evaluation of machine translation. In *StatMT '07: Proceedings of the Second Workshop on Statistical Machine Translation*, pages 136–158, Morristown, NJ, USA. Association for Computational Linguistics.

C. Callison-Burch, T. Cohn, and M. Lapata. 2008. Parametric: An automatic evaluation metric for paraphrasing. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 97–104. Association for Computational Linguistics.

J. Carletta. 1996. Assessing agreement on classification tasks: the kappa statistic. *Computational linguistics*, 22(2):249–254.

Dick Crouch, Mary Dalrymple, Ron Kaplan, Tracy King, John Maxwell, and Paula Newman. 2008. Xle documentation. Technical report, Palo Alto Research Center.

Philip Koenh. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*.

J.R. Landis and G.G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174.

[5] The corpus can be downloaded from `http://www.ling.ohio-state.edu/~mwhite/data/emnlp10/`.

Chin-Yew Lin and Franz Josef Och. 2004. Orange: a method for evaluating automatic evaluation metrics for machine translation. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 501, Morristown, NJ, USA. Association for Computational Linguistics.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. Technical report, IBM Research.

E. Reiter and A. Belz. 2009. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 35(4):529–558.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. III Maxwell, and Mark Johnson. 2002. Parsing the wall street journal using a lexical-functional grammar and discriminative estimation techniques. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 271–278, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

M. Snover, N. Madnani, B.J. Dorr, and R. Schwartz. 2009. Fluency, adequacy, or HTER?: exploring different human judgments with a tunable MT metric. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 259–268. Association for Computational Linguistics.

Amanda Stent, Matthew Marge, and Mohit Singhai. 2005. Evaluating evaluation methods for generation in the presence of variation. In *Proceedings of CICLing*.

J.P. Turian, L. Shen, and I.D. Melamed. 2003. Evaluation of machine translation and its evaluation. *recall (C— R)*, 100:2.

Michael White and Rajakrishnan Rajkumar. 2009. Perceptron reranking for CCG realization. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 410–419, Singapore, August. Association for Computational Linguistics.

Michael White. 2006. Efficient Realization of Coordinate Structures in Combinatory Categorial Grammar. *Research on Language and Computation*, 4(1):39–75.

Type	System	#Refs	#Paraphrases	Average Paraphrases/Ref	#Exact Matches	Adq	Flu
Single output	OpenCCG Baseline 1	296	296	1.0	72	4.17	3.65
	OpenCCG Baseline 2	296	296	1.0	82	4.34	3.94
	OpenCCG Baseline 3	296	296	1.0	76	4.31	3.86
	OpenCCG Perceptron Best	296	1.0	1.0	112	4.37	4.09
	OpenCCG Perceptron Worst	117	117	1.0	5	4.34	3.36
	XLE Best	154	154	1.0	24	4.41	4.07
	XLE Worst	157	157	1.0	13	4.08	3.73
Multiple output	OpenCCG-Perceptron All	296	767	2.6	158	4.45	3.91
	OpenCCG All	296	1131	3.8	162	4.20	3.61
	XLE All	174	557	3.2	54	4.17	3.81
	Wordnet Subsitutions	162	486	3.0	0	3.66	4.71
	Realizer All	296	1628	5.0	169	4.16	3.63
	All	296	2114	7.1	169	4.05	3.88

Table 1: Descriptive statistics

System	Adq			Flu		
	p(A)	p(E)	κ	p(A)	p(E)	κ
OpenCCG-Abs	0.73	0.47	0.48	0.70	0.24	0.61
OpenCCG-Rel	0.76	0.47	0.54	0.76	0.34	0.64
XLE-Abs	0.68	0.42	0.44	0.69	0.27	0.58
XLE-Rel	0.73	0.45	0.50	0.69	0.37	0.50
Wordnet-Abs	0.57	0.25	0.43	0.77	0.66	0.33
Wordnet-Rel	0.74	0.34	0.61	0.73	0.60	0.33
Realizer-Abs	0.70	0.44	0.47	0.69	0.24	0.59
Realizer-Rel	0.74	0.41	0.56	0.73	0.33	0.60
All-Abs	0.67	0.38	0.47	0.71	0.29	0.59
All-Rel	0.74	0.36	**0.60**	0.75	0.34	**0.63**

Table 2: Corpora-wise inter-annotator agreement (absolute and relative κ values shown)

Sys	N	B	M	G	TP	TA	T	HT	HN	HB	HM	HG
OpenCCG-Adq	0.27	*0.39*	0.35	0.18	*0.39*	0.34	**0.4**	**0.43**	0.3	**0.43**	**0.43**	0.23
OpenCCG-Flu	0.49	0.55	0.4	0.42	**0.6**	0.46	**0.6**	**0.72**	0.58	0.69	0.57	0.53
XLE-Adq	*0.52*	*0.51*	**0.55**	0.31	*0.5*	*0.5*	*0.5*	0.52	0.47	0.51	**0.61**	0.4
XLE-Flu	**0.56**	**0.56**	0.48	0.37	*0.55*	0.5	*0.55*	**0.61**	0.54	**0.61**	0.51	0.51
Wordnet-Adq	0.17	0.14	0.24	0.15	**0.37**	0.26	0.22	**0.64**	0.52	0.56	0.32	0.6
Wordnet-Flu	0.26	0.21	0.24	0.24	0.22	**0.27**	0.26	**0.34**	0.32	**0.34**	0.3	**0.34**
Realizer-Adq	0.47	**0.6**	*0.57*	0.42	*0.59*	*0.57*	**0.6**	*0.62*	0.49	*0.62*	**0.65**	0.48
Realizer-Flu	0.51	*0.62*	0.52	0.5	*0.63*	0.53	**0.64**	**0.75**	0.59	*0.73*	0.65	0.63
All-Adq	0.37	0.37	0.33	0.32	*0.42*	0.31	**0.43**	**0.53**	0.44	0.48	0.44	0.45
All-Flu	0.21	**0.62**	0.51	0.32	*0.61*	0.55	*0.6*	*0.7*	0.33	**0.71**	0.62	0.48

Table 3: Spearman's correlations among NIST (N), BLEU (B), METEOR (M), GTM (G), TERp (TP), TERpa (TA), TER (T), human variants (HN, HB, HM, HT, HG) and human judgments (-Adq: adequacy and -Flu: Fluency); Scores which fall within the 95 %CI of the best are italicized.

Sys	N	B	M	G	TP	TA	T	HT	HN	HB	HM	HG
OpenCCG	0.49	0.57	0.42	0.4	0.61	0.46	**0.62**	**0.73**	0.58	0.7	0.59	0.51
XLE	0.63	**0.64**	0.59	0.39	0.62	0.58	0.63	**0.69**	0.6	0.68	0.63	0.54
Wordnet	0.21	0.14	0.21	0.19	**0.38**	0.25	0.23	**0.65**	0.56	0.57	0.31	0.63
Realizer	0.55	0.68	0.57	0.5	0.68	0.58	**0.69**	**0.78**	0.61	0.77	0.7	0.63
All	0.34	0.58	0.47	0.38	**0.61**	0.48	**0.61**	**0.75**	0.48	0.73	0.61	0.58

Table 4: Spearman's correlations among NIST (N), BLEU (B), METEOR (M), GTM (G), TERp (TP), TERpa (TA), TER (T), human variants (HN, HB, HM, HT, HG) and human judgments (combined adequacy and fluency scores)

System	Adq			Flu		
	Sp	95%L	95%U	Sp	95%L	95%U
Realizer	0.60	0.58	0.63	0.62	0.59	0.65
XLE	0.51	0.47	0.56	0.56	0.51	0.61
OpenCCG	0.39	0.35	0.42	0.55	0.52	0.59
All	0.37	0.34	0.4	0.62	0.6	0.64
Wordnet	0.14	0.06	0.21	0.21	0.13	0.28

Table 5: Spearman's correlation analysis (bootstrap sampling) of the BLEU scores of various systems with human adequacy and fluency scores

Sys	HJ	N	B	M	G	TP	TA	T	HT	HN	HB	HM	HG	HJ1-HJ2
OpenCCG	HJ-1	0.44	0.52	0.39	0.36	0.56	0.43	**0.58**	**0.75**	0.58	0.72	0.62	0.52	0.76
	HJ-2	0.5	0.58	0.43	0.4	0.62	0.46	**0.63**	**0.7**	0.55	0.68	0.56	0.49	
XLE	HJ-1	**0.6**	**0.6**	0.55	0.37	0.57	0.55	0.58	**0.69**	0.63	0.68	0.64	0.54	0.75
	HJ-2	0.6	0.6	0.56	0.39	0.6	0.55	**0.61**	**0.64**	0.54	0.61	0.57	0.51	
Wordnet	HJ-1	0.2	0.18	0.26	0.16	**0.37**	0.28	0.24	**0.7**	0.59	0.64	0.35	0.65	0.72
	HJ-2	0.25	0.16	0.23	0.19	**0.37**	0.25	0.25	**0.59**	0.52	0.51	0.32	0.56	
Realizer	HJ-1	0.51	0.65	0.56	0.49	0.64	0.56	**0.66**	**0.8**	0.62	0.78	0.72	0.64	0.82
	HJ-2	0.55	**0.68**	0.56	0.5	0.67	0.57	**0.68**	**0.74**	0.58	0.73	0.66	0.6	
All	HJ-1	0.32	0.53	0.45	0.37	**0.57**	0.44	**0.57**	**0.77**	0.5	0.74	0.62	0.59	0.79
	HJ-2	0.35	0.58	0.46	0.37	**0.61**	0.47	0.6	**0.71**	0.44	0.69	0.57	0.54	

Table 6: Spearman's correlations of NIST (N), BLEU (B), METEOR (M), GTM (G), TERp (TP), TERpa (TA), human variants (HT, HN, HB, HM, HG), and individual human judgments (combined adq. and flu. scores)

Factor	*Count*
Punctuation	17
Adverbial shift	16
Agreement	14
Other shifts	8
Conjunct rearrangement	8
Complementizer ins/del	5
PP shift	4

Table 7: Factors influencing TERP ranking errors for 50 worst-ranked realization groups

Metric	Score	Errors
nist	4	C1-XB
bleu	3	XB-PW C1-XB
meteor	4	XW-PB
ter	2	PW-PB XW-PB C1-PB
terp	2	PW-PB XW-PB C1-PB
terpa	3	XW-PB C1-PB
gtm	4	C1-XB
hnist	5	
hbleu	3	PW-PB XW-PB
hmeteor	2	PW-PB XW-PB C1-PB
hter	3	XB-PW C1-XB
hgtm	3	XB-PW C1-XB

Table 8: Metric-wise ranking performance in terms of agreement with a ranking induced by combined adequacy and fluency scores; each metric gets a score out of 5 (i.e. number of system-level comparisons that emerged significant as per the Tukey's HSD test)
Legend: Perceptron Best (PB); Perceptron Worst (PW); XLE Best (XB); XLE Worst (XW); OpenCCG baseline models 1 to 3 (C1 ... C3)

Two Decades of Unsupervised POS induction: How far have we come?

Christos Christodoulopoulos
School of Informatics
University of Edinburgh
christos.c@ed.ac.uk

Sharon Goldwater
School of Informatics
University of Edinburgh
sgwater@inf.ed.ac.uk

Mark Steedman
School of Informatics
University of Edinburgh
steedman@inf.ed.ac.uk

Abstract

Part-of-speech (POS) induction is one of the most popular tasks in research on unsupervised NLP. Many different methods have been proposed, yet comparisons are difficult to make since there is little consensus on evaluation framework, and many papers evaluate against only one or two competitor systems. Here we evaluate seven different POS induction systems spanning nearly 20 years of work, using a variety of measures. We show that some of the oldest (and simplest) systems stand up surprisingly well against more recent approaches. Since most of these systems were developed and tested using data from the WSJ corpus, we compare their generalization abilities by testing on both WSJ and the multilingual Multext-East corpus. Finally, we introduce the idea of evaluating systems based on their ability to produce cluster prototypes that are useful as input to a prototype-driven learner. In most cases, the prototype-driven learner outperforms the unsupervised system used to initialize it, yielding state-of-the-art results on WSJ and improvements on non-English corpora.

1 Introduction

In recent years, unsupervised learning has become a hot area in NLP, in large part due to the use of sophisticated machine learning approaches which promise to deliver better results than more traditional methods. Often the new approaches are tested using part-of-speech (POS) tagging as an example application, and usually they are shown to perform better than one or another comparison system. However, it is difficult to draw overall conclusions about the relative performance of unsupervised POS tagging systems because of differences in evaluation measures, and the fact that no paper includes direct comparisons against more than a few other systems. In this paper, we attempt to remedy that situation by providing a comprehensive evaluation of seven different POS induction systems spanning nearly 20 years of research. We focus specifically on POS *induction* systems, where no prior knowledge is available, in contrast to POS *disambiguation* systems (Merialdo, 1994; Toutanova and Johnson, 2007; Naseem et al., 2009; Ravi and Knight, 2009; Smith and Eisner, 2005), which use a dictionary to provide possible tags for some or all of the words in the corpus, or *prototype-driven* systems (Haghighi and Klein, 2006), which use a small set of prototypes for each tag class, but no dictionary. Our motivation stems from another part of our own research, in which we are trying to use NLP systems on over 50 low-density languages (some of them dead) where both tagged corpora and language speakers are mostly unavailable. We therefore desire to use these systems straight out of the box and to know how well we can expect them to work.

One difficulty in evaluating POS induction systems is that there is no straightforward way to map the clusters found by the algorithm onto the gold standard tags; moreover, some systems are designed to induce the number of clusters as well as their contents, so the number of found clusters may not match either the gold standard or that of another system. Nevertheless, most recent papers have used mapping-based performance measures (either *one-to-one* or *many-to-one* accuracy). Here, we argue that the entropy-based V-Measure (Rosenberg and

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 575–584,
MIT, Massachusetts, USA, 9-11 October 2010.

Hirschberg, 2007) is more useful in many cases, being more stable across different numbers of found and true clusters, and avoiding several of the problems with another commonly used entropy-based measure, Variation of Information (Meilă, 2003).

Using V-Measure along with several other evaluation measures, we compare the performance of the different induction systems on both WSJ (the data on which most systems were developed and tested) and Multext East, a corpus of parallel texts in eight different languages. We find that for virtually all measures and datasets, older systems using relatively simple models and algorithms (Brown et al., 1992; Clark, 2003) work as well or better than systems using newer and often far more sophisticated and time-consuming machine learning methods (Goldwater and Griffiths, 2007; Johnson, 2007; Graca et al., 2009; Berg-Kirkpatrick et al., 2010). Thus, although these newer methods have introduced potentially useful machine learning techniques, they should not be assumed to provide the best performance for unsupervised POS induction.

In addition to our review and comparison, we introduce a new way to both evaluate and potentially improve a POS induction system. Our method is based on the prototype-driven learning system of Haghighi and Klein (2006), which achieves very good performance by using a hand-selected list of prototypes for each syntactic cluster. We instead use the existing POS induction systems to induce prototypes automatically, and evaluate the systems based on the quality of their prototypes. We find that the oldest system tested (Brown et al., 1992) produces the best prototypes, and that using these prototypes as input to Haghighi and Klein's system yields state-of-the-art performance on WSJ and improvements on seven of the eight non-English corpora.

2 POS Induction Systems

We describe each system only briefly; for details, see the respective papers, cited below. Each system outputs a set of syntactic clusters C; except where noted, the target number of clusters $|C|$ must be specified as an input parameter. Since we are interested in out-of-the-box performance, we use the default parameter settings for each system, except for $|C|$, which is varied in some of our experiments. The systems are as follows:[1]

[brown]: Class-based n-grams (Brown et al., 1992). This is the oldest and one of the simplest systems we tested. It uses a bigram model where each word type is assigned to a latent class (a hard assignment), and the probability of the corpus $w_1 \dots w_n$ is computed as $P(w_1|c_1) \prod_{i=2}^{n} P(w_i|c_i)P(c_i|c_{i-1})$, where c_i is the class of w_i. The goal is to optimize the probability of the corpus under this model. The authors use an approximate search procedure: greedy agglomerative hierarchical clustering followed by a step in which individual word types are considered for movement to a different class if this improves the corpus probability.

[clark]: Class-based n-grams with morphology (Clark, 2003). This system uses a similar model to the previous one, and also clusters word types (rather than tokens, as the rest of the systems do). The main differences between the systems are that clark uses a slightly different approximate search procedure, and that he augments the probabilistic model with a prior that prefers clusterings where morphologically similar words are clustered together. The morphology component is implemented as a single-order letter HMM.

[cw]: Chinese Whispers graph clustering (Biemann, 2006). Unlike the other systems we consider, this one induces the value of $|C|$ rather than taking it as an input parameter.[2] The system uses a graph clustering algorithm called *Chinese Whispers* that is based on contextual similarity. The algorithm works in two stages. The first clusters the most frequent 10,000 words (*target words*) based on their context statistics, with contexts formed from the most frequent 150-250 words (*feature words*) that appear ei-

[1] Implementations were obtained from:
brown: `http://www.cs.berkeley.edu/~pliang/software/brown-cluster-1.2.zip` (Percy Liang),
clark: `http://www.cs.rhul.ac.uk/home/alexc/pos2.tar.gz` (Alex Clark),
cw: `http://wortschatz.uni-leipzig.de/%7Ecbiemann/software/jUnsupos1.0.zip` (Chris Biemann),
bhmm, **vbhmm**, **pr**, **feat**: by request from the authors of the respective papers.

[2] Another recent model that induces $|C|$ is the Infinite HMM (Van Gael et al., 2009). Unfortunately, we were unable to obtain code for the IHMM in time to include it in our analysis. Van Gael et al. (2009) report results of around 59% V-Measure on WSJ, with 194 induced clusters, which is not as good as the best system scores in Section 4.

ther to the left or right of a target word. The second stage deals with medium and low frequency words and uses pairwise similarity scores calculated by the number of shared neighbors between two words in a 4-word context window. The final clustering is a combination of the clusters obtained in the two stages. While the number of target words, feature words, and window size are in principle parameters of the algorithm, they are hard-coded in the implementation we used and we did not change them.

[bhmm]: Bayesian HMM with Gibbs sampling (Goldwater and Griffiths, 2007). This system is based on a standard HMM for POS tagging. It differs from the standard model by placing Dirichlet priors over the multinomial parameters defining the state-state and state-emission distributions, and uses a collapsed Gibbs sampler to infer the hidden tags. The Dirichlet hyperparameters α (which controls the sparsity of the transition probabilities) and β (which controls the sparsity of the emission probabilities) can be fixed or inferred. We used a bigram version of this model with hyperparameter inference.

[vbhmm]: Bayesian HMM with variational Bayes (Johnson, 2007). This system uses the same bigram model as **bhmm**, but uses variational Bayesian EM for inference. We fixed the α and β parameters to 0.1, values that appeared to be reasonable based on Johnson (2007), and which were also used by Graca et al. (2009).

[pr]: Sparsity posterior-regularization HMM (Graca et al., 2009). The Bayesian approaches described above encourage sparse state-state and state-emission distributions only indirectly through the Dirichlet priors. This system, while utilizing the same bigram HMM, encourages sparsity directly by constraining the *posterior* distributions using the posterior regularization framework (Ganchev et al., 2009). A parameter σ controls the strengths of the constraints (default = 25). Following Graca et al. (2009), we set $\alpha = \beta = 0.1$.

[feat]: Feature-based HMM (Berg-Kirkpatrick et al., 2010). This system uses a model that has the structure of a standard HMM, but assumes that the state-state and state-emission distributions are logistic, rather than multinomial. The logistic distributions allow the model to incorporate local features of the sort often used in discriminative models. The default features are morphological, such as character trigrams and capitalization.

3 Evaluation Measures

One difficulty in comparing POS induction methods is in finding an appropriate evaluation measure. Many different measures have been proposed over the years, but there is still no consensus on which is best. In addition, some measures with supposed theoretical advantages, such as Variation of Information (VI) (Meilă, 2003) have had little empirical analysis. Our goal in this section is to determine which of these measures is most sensible for evaluating the systems presented above. We first describe each measure before presenting empirical results. Except for VI, all measures range from 0 to 1, with higher scores indicating better performance.

[many-to-1]: Many-to-one mapping accuracy (also known as *cluster purity*) maps each cluster to the gold standard tag that is most common for the words in that cluster (henceforth, the *preferred tag*), and then computes the proportion of words tagged correctly. More than one cluster may be mapped to the same gold standard tag. This is the most commonly used metric across the literature as it is intuitive and creates a meaningful POS sequence out of the cluster identifiers. However, it tends to yield higher scores as $|C|$ increases, making comparisons difficult when $|C|$ can vary.

[crossval]: Cross-validation accuracy (Gao and Johnson, 2008) is intended to address the problem with many-to-one accuracy which is that assigning each word to its own class yields a perfect score. In this measure, the first half of the corpus is used to obtain the many-to-one mapping of clusters to tags, and this mapping is used to compute the accuracy of the clustering on the second half of the corpus.

[1-to-1]: One-to-one mapping accuracy (Haghighi and Klein, 2006) constrains the mapping from clusters to tags, so that at most one cluster can be mapped to any tag. The mapping is performed greedily. In general, as the number of clusters increases, fewer clusters will be mapped to their preferred tag and scores will decrease (especially if the number of clusters is larger than the number of tags, so that some clusters are unassigned and receive zero credit). Again, this makes it difficult to compare solutions with different values of $|C|$.

[vi]: Variation of Information (Meilă, 2003) is an information-theoretic measure that regards the system output C and the gold standard tags T as two separate clusterings, and evaluates the amount of information lost in going from C to T and the amount of information gained, i.e., the sum of the conditional entropy of each clustering conditioned on the other. More formally, $VI(C,T) = H(T|C) + H(C|T) = H(C) + H(T) - 2I(C,T)$, where $H(.)$ is the entropy function and $I(.)$ is the mutual information. VI and other entropy-based measures have been argued to be superior to accuracy-based measures such as those above, because they consider not only the majority tag in each cluster, but also whether the remainder of the cluster is more or less homogeneous. Unlike the other measures we consider, lower scores are better (since VI measures the difference between clusterings in bits).

[vm]: V-Measure (Rosenberg and Hirschberg, 2007) is another entropy-based measure that is designed to be analogous to F-measure, in that it is defined as the weighted harmonic mean of two values, homogeneity (h, the precision analogue) and completeness (c, the recall analogue):

$$h = 1 - \frac{H(T|C)}{H(T)} \quad (1)$$

$$c = 1 - \frac{H(C|T)}{H(C)} \quad (2)$$

$$VM = \frac{(1+\beta)hc}{(\beta h) + c} \quad (3)$$

As with F-measure, β is normally set to 1.

[vmb]: V-beta is an extension to V-Measure, proposed by (Vlachos et al., 2009). They noted that V-Measure favors clusterings where the number of clusters $|C|$ is larger than the number of POS tags $|T|$. To address this issue the parameter β in equation 3 is set to $|C|/|T|$ in order adjust the balance between homogeneity and completeness.

[s-fscore]: Substitutable F-score (Frank et al., 2009). One potential issue with all of the above measures is that they require a gold standard tagging to compute. This is normally available during development of a system, but if the system is deployed on a novel language a gold standard may not be available. In addition, there is the question of whether the gold standard itself is "correct". Recently, Frank et al. (2009) proposed this novel evaluation measure that requires no gold standard, instead using the concept of substitutability to evaluate performance. Instead of comparing the system's clusters C to gold standard clusters T, they are compared to a set of clusters S created from *substitutable frames*, i.e., clusters of words that occur in the same syntactic environment. Ideally a substitutable frame would be created by sentences differing in only one word (e.g. "I want the blue ball." and "I want the red ball.") and the resulting cluster would contain the words that change (e.g. [blue, red]). However since it is almost impossible to find these types of sentences in real-world corpora, the authors use frames created by two words appearing in the corpus with exactly one word between (e.g. the — ball). Once the substitutable clusters have been created, they can be used to calculate the Precision (SP), Recall (SR) and F-score (SF) of the system's clustering:

$$SP = \frac{\sum_{s\in S}\sum_{c\in C}|s\cap c|(|s\cap c|-1)}{\sum_{c\in C}|c|(|c|-1)} \quad (4)$$

$$SR = \frac{\sum_{s\in S}\sum_{c\in C}|s\cap c|(|s\cap c|-1)}{\sum_{s\in S}|s|(|s|-1)} \quad (5)$$

$$SF = \frac{2\cdot SP\cdot SR}{SP+SR} \quad (6)$$

3.1 Empirical results

We mentioned a few strengths and weaknesses of each evaluation method above; in this section we present some empirical results to expand on these claims. First, we examine the effects of varying $|C|$ on the behavior of the evaluation measures, while keeping the number of gold standard tags the same ($|T| = 45$). Results were obtained by training and evaluating each system on the full WSJ portion of the Penn Treebank corpus (Marcus et al., 1993). Figure 1 shows the results from the Brown system for $|C|$ ranging from 1 to 200; the same trends were observed for all other systems.[3] In addition, Table 1 provides results for the two extremes of $|C| = 1$ (**all** words assigned to the same cluster) and $|C|$ equal to the size of the corpus (a **single** word per cluster), as

[3] The results reported in this paper are only a fraction of the total from our experiments; given the number of parameters, models and measures tested, we obtained over 15000 results. The full set of results can be found at `http://homepages.inf.ed.ac.uk/s0787820/pos/`.

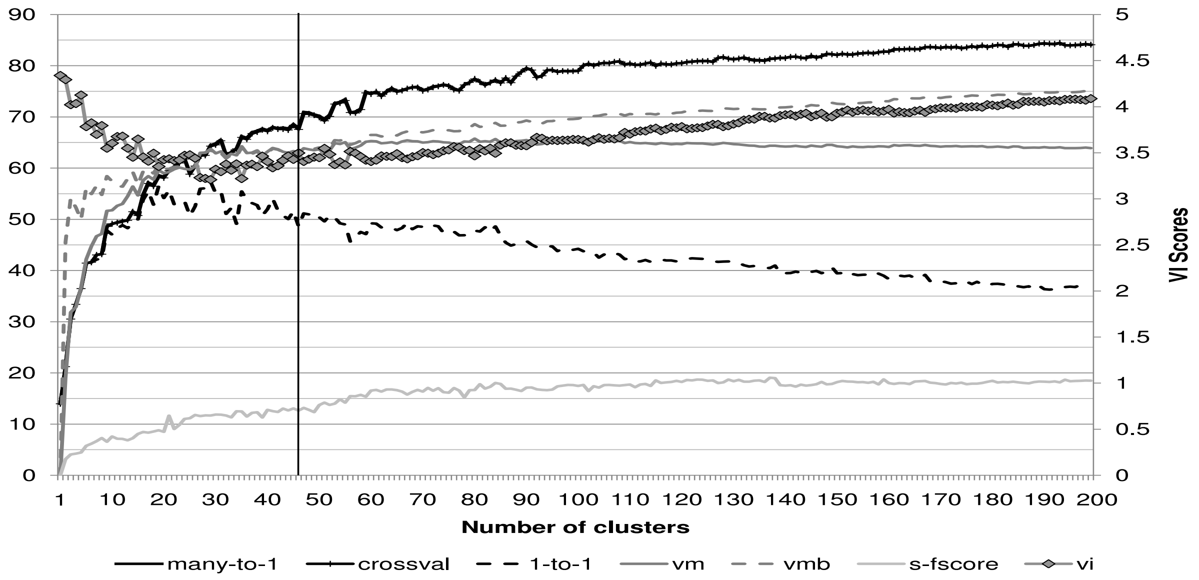


Figure 1: Scores for all evaluation measures as a function of the number of clusters returned [model:brown, corpus:wsj, $|C|$:{1-200}, $|T|$:45]. The right-hand y-axis shows VI scores (lower is better); the left-hand y-axis shows percentage scores for all other measures. The vertical line indicates $|T|$. Many-to-1 is invisible as it tracks crossval so closely.

measure	super	random	all	single
many-to-1	97.85	13.97	13.97	100
crossval	97.59	13.98	13.98	0
1-to-1	97.86	2.42	13.97	0.01
vi	0.35	9.81	4.33	15.82
vm	95.98	0.02	0	35.42
vmb	95.98	0	0	99.99
s-fscore	7.53	0.50	0	0

Table 1: Baseline scores for the different evaluation measures on the WSJ corpus. For all measures except VI higher is better.

well as two other baselines (a **super**vised tagging[4] and a **random** clustering with $|C| = 45$).

These empirical results confirm that certain measures favor solutions with many clusters, while others prefer fewer clusters. As expected, **many-to-1** correlates positively with $|C|$, rising to almost 85% with $|C| = 200$ and reaching 100% when the number of clusters is maximal (i.e., **single**). Recall that **crossval** was proposed as a possible solution to this problem, and it does solve the extreme case of **single**, yielding 0% accuracy rather than 100%. However, it patterns just like **many-to-1** for up to 200 clusters, suggesting that there is very little difference between the two for any reasonable number of clusters, and we should be wary of using either one when $|C|$ may vary.

In contrast to these measures are **1-to-1** and **vi**: for the most part, they yield worse performance (lower **1-to-1**, higher **vi**) as $|C|$ increases. However, in this case the trend is not monotonic: there is an initial improvement in performance before the decrease begins. One might hope that the peak in performance would occur when the number of clusters is approximately equal to the number of gold standard tags; however, the best performance for both **1-to-1** and **vi** occurs with approximately 25-30 clusters, many fewer than the gold standard 45.

Next we consider **vm** and **vmb**. Interestingly, although **vmb** was proposed as a way to correct for the supposed tendency of **vm** to increase with increasing $|C|$, we find that **vm** is actually more stable than **vmb** over different values of $|C|$. Thus, if the goal is to compare systems producing different numbers of clusters (especially important for systems that induce the number of clusters), then **vm** seems more appropriate than any of the above measures, which are more standard in the literature.

Finally, we analyze the behavior of the gold-standard-independent measure, **s-fscore**. On the positive side, this measure assigns scores of 0 to the

[4] We used the Stanford Tagger trained on the WSJ corpus: `http://nlp.stanford.edu/software/tagger.shtml`.

two extreme cases of **all** and **single**, and is relatively stable across different values of $|C|$ after an initial increase. It assigns a lower score to the supervised system than to **brown**, indicating that words in the supervised clusters (which are very close to the gold standard) are actually less substitutable than words in the unsupervised clusters. This is probably due to the fact that the gold standard encodes “pure” syntactic classes, while substitutability also depends on semantic characteristics (which tend to be picked up by unsupervised clustering systems as well). Another potential problem with this measure is that it has a very small dynamic range – while scores as high as 1 are theoretically possible, in practice they will never be achieved, and we see that the actual range of scores observed are all under 20%.

We conclude that there is probably no single evaluation measure that is best for all purposes. If a gold standard is available, then **many-to-1** and **1-to-1** are the most intuitive measures, but should not be used when $|C|$ is variable, and do not account for differences in the errors made. While **vi** has been popular as an entropy-based alternative to address the latter problem, its scores are not easy to interpret (being on a scale of bits) and it still has the problem of incomparability across different $|C|$. Overall, **vm** seems to be the best general-purpose measure that combines an entropy-based score with an intuitive 0-1 scale and stability over a wide range of $|C|$.

4 System comparison

Having provided some intuitions about the behavior of different evaluation methods, we move on to evaluating the various systems presented in Section 2. We first present results for the same WSJ corpus used above. However, because most of the systems were initially developed on this corpus, and often evaluated only on it, there is a question of whether their methods and/or hyperparameters are overly specific to the domain or to the English language. This is a particularly pertinent question since a primary argument in favor of unsupervised systems is that they are easier to port to a new language or domain than supervised systems. To address this question, we evaluate all the systems as well on the multilingual Multext East corpus (Erjavec, 2004), without changing any of the parameter settings. $|C|$ was set to 45 for all of the experiments reported in this section. Based on our assessment of evaluation measures above, we report VM scores as the most reliable measure across different systems and cluster set sizes; to facilitate comparisons with previous papers, we also report many-to-one and one-to-one accuracy.

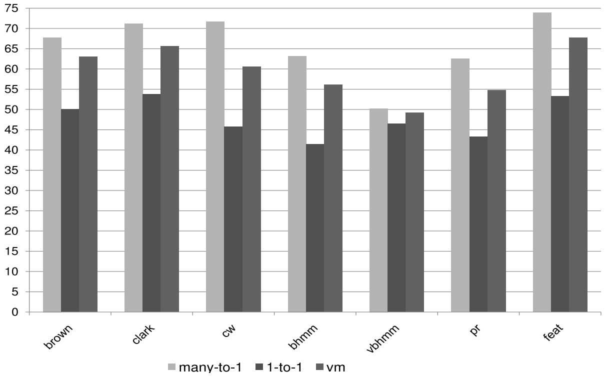


Figure 2: Performance of the different systems on WSJ, using three different measures [$|C|$:45, $|T|$:45]

system	runtime
brown	~10 min.
clark	~40 min.
cw	~10 min.
bhmm	~4 hrs.
vbhmm	~10 hrs.
pr	~10 hrs.*
feat	~40 hrs.*

Table 2: Runtimes for the different systems on WSJ [$|C|$:45]. ***pr** and **feat** have multithreading implementations and ran on 16 cores.

4.1 Results on WSJ

Figure 2 presents results for all seven systems, with approximate runtimes shown in Table 2. While these algorithms have not necessarily been optimized for speed, there is a fairly clear distinction between the older type-clustering models (**brown**, **clark**) and the graph-based algorithm (**cw**) on the one hand, and the newer machine-learning approaches (**bhmm**, **vbhmm**, **pr**, **feat**) on the other, with the former being much faster to run. Despite their faster runtimes and less sophisticated methods, however, these systems perform surprisingly well in comparison to the latter group. Even the oldest and perhaps simplest method (**brown**) outperforms the two BHMMs and posterior regularization on all measures. Only

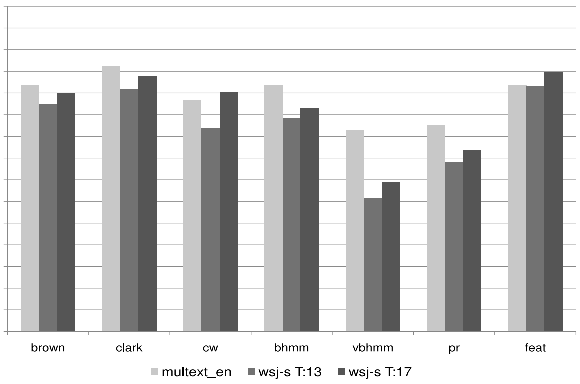


Figure 3: VM scores for the different systems on English Multext-East and WSJ-S corpora [$|C|$:45, $|T|$:{14,17}]

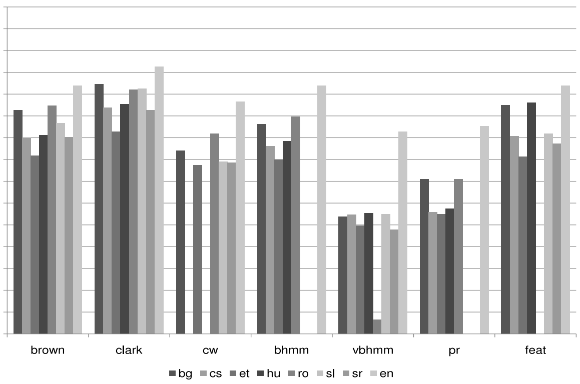


Figure 4: VM scores for the different systems on the eight Multext-East corpora [$|C|$:45, $|T|$:14]

the very latest approach (**feat**) rivals **clark**, showing slightly better performance on two of the three measures (**clark**: 71.2, 53.8, 65.5 on many-to-one, one-to-one, VM; **feat**: 73.9, 53.3, 67.7). The **cw** system returns a total of 568 clusters on this data set, so the many-to-one and one-to-one measures are not strictly comparable to the other systems; on VM this system achieves middling performance.

We note that the two best-performing systems, **clark** and **feat**, are also the only two to use morphological information. Since the clustering algorithms used by **brown** and **clark** are quite similar, the difference in performance between the two can probably be attributed to the extra information provided by the morphology. This supports the (unsurprising) conclusion that incorporating morphological features is generally helpful for POS induction.

4.2 Results on other corpora

We now examine whether either the relative or absolute performance of the different systems holds up when tested on a variety of different languages. For these experiments, we used the 1984 portion of the Multext-East corpus (~7k sentences), which contains parallel translations of Orwell's *1984* in 8 different languages: Bulgarian[**bg**], Czech[**cs**], Estonian[**et**], Hungarian[**hu**], Romanian[**ro**], Slovene[**sl**], Serbian[**sr**] and English[**en**]. We also included a 7k sentence version of the WSJ corpus [**wsj-s**] to help differentiate effects of corpus size from those of domain/language. For the WSJ corpora we experimented with two standardly used tagsets: the original PTB 45-tag gold standard and a coarser set of 17 tags previously used by several researchers working on unsupervised POS tagging (Smith and Eisner, 2005; Goldwater and Griffiths, 2007; Johnson, 2007). For the Multext-East corpus only a coarse 14-tag tagset was available.[5] Finally, to facilitate direct comparisons of genre while controlling for the size of both the corpus and the tag set, we also created a further collapsed 13-tag set for WSJ.[6]

Figure 3 illustrates the abilities of the different systems to generalize across different genres of English text. Comparing the results for the Multext-East English corpus and the small WSJ corpus with 13 tags (i.e., controlling as much as possible for corpus size and number of gold standard tags), we see that despite being developed on WSJ, the systems actually perform better on Multext-East. This is encouraging, since it suggests that the methods and hyperparameters of the algorithms are not strongly tied to WSJ. It also suggests that Multext-East is in some sense an easier corpus than WSJ. Indeed, the distribution of vocabulary items supports this view: the 100 most frequent words account for 48% of the WSJ corpus, but 57% of the 1984 novel. It is also worth pointing out that, although previous researchers have reduced the 45-tag WSJ set to 17 tags in order to create an easier task for unsupervised learning (and to decrease training time), reducing the tag set further to 13 tags actually decreases performance, since some distinctions found by the sys-

[5] Out of the 14 tags only 11 are shared across all languages. For details c.f. Appendix B in (Naseem et al., 2009).

[6] We tried to make the meanings of the tags as similar as possible between the two corpora; we had to create 13 rather than 14 WSJ tags for this reason. Our 13-tag set can be found at `http://homepages.inf.ed.ac.uk/s0787820/pos/`.

tems (e.g., between different types of punctuation) are collapsed in the gold standard.

Figure 4 gives the results of the different systems on the various languages.[7] Not surprisingly, all the algorithms perform best on English, often by a wide margin, suggesting that they are indeed tuned better towards English syntax and/or morphology. One might expect that the two systems with morphological features (**clark** and **feat**) would show less difference between English and some of the other languages (all of which have complex morphology) than the other systems. However, although **clark** and **feat** (along with Brown) are the best performing systems overall, they don't show any particular benefit for the morphologically complex languages.[8]

One difference between the Multext-East results and the WSJ results is that on Multext-East, **clark** clearly outperforms all the other systems. This is true for both the English and non-English corpora, despite the similar performance of **clark** and **feat** on (English) WSJ. This suggests that **feat** benefits more from the larger corpus size of WSJ. For the other languages **clark** may be benefiting from somewhat more general morphological features; **feat** currently contains suffix features but no prefix features (although these could be added).

Overall, our experiments on multiple languages support our earlier claim that many of the newer POS induction systems are not as successful as the older methods. Moreover, these experiments underscore the importance of testing unsupervised systems on multiple languages and domains, since both the absolute and relative performance of systems may change on different data sets. Ideally, some of the corpora should be held out as unseen test data if an effective argument is to be made regarding the language- or domain-generality of the system.

5 Learning from induced prototypes

We now introduce a final novel method of evaluating POS induction systems and potentially improving their performance as well. Our idea is based on the prototype-driven learning model of Haghighi and Klein (2006). This model is unsupervised, but requires as input a handful of *prototypes* (canonical examples) for each word class. The system uses a log-linear model with features that include the prototype lists as well as morphological features (the same ones used in **feat**). Using the most frequent words in each gold standard class as prototypes, the authors report 80.5% accuracy (both many-to-one and one-to-one) on WSJ, considerably higher than any of the induction systems seen here. This raises two questions: If we wish to induce prototypes without a tagged corpus or language-specific knowledge, which induction system will provide the best prototypes (i.e., most similar to the gold standard prototypes)? And, can we use the induced prototypes as input to the prototype-driven model (**h&k**) to achieve better performance than the system the prototypes were extracted from?

To explore these questions, we implemented a simple heuristic method for inducing prototypes from the output C of a POS induction system by selecting a few frequent words in each cluster that are the most similar to other words in the cluster and also the most dissimilar to the words in other clusters. For each cluster $c_i \in C$, we retain as candidate prototypes the words whose frequency in c_i is at least 90% as high as the word with the highest frequency (in c_i). This yields about 20-30 candidates from each cluster. For each of these, we compute its average similarity $\mathcal{S}$ to the other candidates in its cluster, and the average dissimilarity $\mathcal{D}$ to the candidates in other clusters. Similarity is computed using the method described by Haghighi and Klein (2006), which uses SVD on word context vectors and cosine similarity. Dissimilarity between a pair of words is computed as one minus the similarity. Finally we compute the average $\mathcal{M} = 0.5(\mathcal{S} + \mathcal{D})$, sort the words by their $\mathcal{M}$ scores, and keep as prototypes the top ten words with $\mathcal{M} > 0.25 * \max_{c_i}(\mathcal{M})$. The cutoff threshold results in some clusters having less than ten prototypes, which is appropriate since some gold standard categories have very few members (e.g., punctuation, determiners).

Using this method, we first tested the various base+proto systems on the WSJ corpus. Results in Table 3 show that the **brown** system produces the best prototypes. Although not as good as using prototypes from the gold standard (**h&k**),

[7] Some results are missing because not all of the corpora were successfully processed by all of the systems.

[8] It can be argued that lemmatization would have given a significant gain to the performance of the systems in these languages. Although lemmatization information was included in the corpus we chose not to use it, maintaining the fully unsupervised nature of this task.

system	many-to-1	1-to-1	vm
brown	**76.1**(8.3)	60.7(10.6)	**68.8**(5.8)
clark	74.5(3.3)	62.1(8.3)	68.6(3.0)
bhmm	71.8(8.6)	56.5(15.0)	65.7(9.5)
vbhmm	68.1(**17.9**)	**67.2**(**20.7**)	67.5(**18.3**)
pr	71.6(9.2)	60.2(17.0)	67.2(12.4)
feat	69.8(-4.1)	52.0(-1.3)	63.1(-4.6)
h&k	80.2	80.2	75.2

Table 3: Scores on WSJ for our prototype-based POS induction system, with prototypes extracted from each of the existing systems [$|C|$:45,$|T|$:45]. Numbers in parentheses are the improvement over the same system without using the prototype step. Scores in bold indicate the best performance (improvement) in each column. **h&k** uses gold standard prototypes.

corpus	**brown**	**clark**
wsj	68.8(5.8)	68.5(3.0)
wsj-s	62.3(2.7)	67.5(3.6)
en	58.5(1.6)	57.9(-3.3)
bg	53.7(2.3)	50.2(-7.1)
cs	49.9(5.0)	48.0(-4.0)
et	45.8(4.9)	44.4(-1.9)
hu	45.8(0.1)	47.0(-5.7)
ro	53.2(0.8)	52.7(-3.3)
sl	51.2(2.9)	51.7(-4.6)
sr	48.0(2.8)	46.4(-4.9)

Table 4: VM scores for **brown+proto** and **clark+proto** on all corpora. Numbers in parentheses indicate improvement over the base systems.

brown+proto yields a large improvement over **brown**, and the highest performance of any system tested so far. In fact, the **brown+proto** scores are, to our knowledge, the best reported results for an unsupervised POS induction system on WSJ.

Next, we evaluated the two best-performing **+proto** systems on Multext-East, as shown in Table 4. We see that **brown** again yields the best prototypes, and again yields improvements when used as **brown+proto** (although the improvements are not as large as those on WSJ). Interestingly, **clark+proto** actually performs worse than **clark** on the multilingual data, showing that although induced prototypes can in principle improve the performance of a system, not all systems will benefit in all situations. This suggests a need for additional investigation to determine what properties of an existing induction system allow it to produce useful prototypes with the current method and/or to develop a specialized system specifically targeted towards inducing useful prototypes.

6 Conclusion

In this paper, we have attempted to provide a more comprehensive review and comparison of evaluation measures and systems for POS induction than has been done before. We pointed out that most of the commonly used evaluation measures are sensitive to the number of induced clusters, and suggested that V-measure (which is less sensitive) should be used as an alternative or in conjunction with the standard measures. With regard to the systems themselves, we found that many of the newer approaches actually perform worse than older methods that are both simpler and faster. The newer systems have introduced potentially important machine learning tools, but are not necessarily better suited to the POS induction task specifically.

Since portability is a distinguishing feature for unsupervised models, we have stressed the importance of testing the systems on corpora that were not used in their development, and especially on different languages. We found that on non-English languages, Clark's (2003) system performed best.

Finally, we introduced the idea of evaluating induction systems based on their ability to produce useful cluster prototypes. We found that the oldest system (Brown et al., 1992) yielded the best prototypes, and that using these prototypes gave state-of-the-art performance on WSJ, as well as improvements on nearly all of the non-English corpora. These promising results suggest a new direction for future research: improving POS induction by developing methods targeted towards extracting better prototypes, rather than focusing on improving clustering of the entire data set.

Acknowledgments

We thank Mark Johnson, Kuzman Ganchev, and Taylor Berg-Kirkpatrick for providing the implementations of their models, as well as Stella Frank, Tom Kwiatkowski, Luke Zettlemoyer and the anonymous reviewers for their comments and suggestions. This work was supported by an EPSRC graduate Fellowship, and by ERC Advanced Fellowship 249520 GRAMPLUS.

References

Taylor Berg-Kirkpatrick, Alexandre B. Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proceedings of NAACL 2010*, pages 582–590, Los Angeles, California, June.

Chris Biemann. 2006. Unsupervised part-of-speech tagging employing efficient graph clustering. In *Proceedings of COLING ACL 2006*, pages 7–12, Morristown, NJ, USA.

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.

Alexander Clark. 2003. Combining distributional and morphological information for part of speech induction. In *Proceedings of EACL 2003*, pages 59–66, Morristown, NJ, USA.

Tomaž Erjavec. 2004. MULTEXT-East Version 3: Multilingual Morphosyntactic Specifications, Lexicons and Corpora. In *Fourth International Conference on Language Resources and Evaluation, LREC'04*, page In print, Paris. ELRA.

Stella Frank, Sharon Goldwater, and Frank Keller. 2009. Evaluating models of syntactic category acquisition without using a gold standard. In *Proceedings of CogSci09*, July.

Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2009. Posterior regularization for structured latent variable models. Technical report, University of Pennsylvania.

Jianfeng Gao and Mark Johnson. 2008. A comparison of bayesian estimators for unsupervised hidden markov model pos taggers. In *Proceedings of EMNLP 2008*, pages 344–352, Morristown, NJ, USA.

Sharon Goldwater and Tom Griffiths. 2007. A fully bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of ACL 2007*, pages 744–751, Prague, Czech Republic, June.

Joao Graca, Kuzman Ganchev, Ben Taskar, and Fernando Pereira. 2009. Posterior vs parameter sparsity in latent variable models. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 664–672.

Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *Proceedings of NAACL 2006*, pages 320–327, Morristown, NJ, USA.

Mark Johnson. 2007. Why doesn't EM find good HMM POS-taggers? In *Proceedings of EMNLP-CoNLL 2007*, pages 296–305, Prague, Czech Republic, June.

M. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):331–330.

Marina Meilă. 2003. Comparing clusterings by the variation of information. In *Learning Theory and Kernel Machines*, pages 173–187.

B. Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–172.

Tahira Naseem, Benjamin Snyder, Jacob Eisenstein, and Regina Barzilay. 2009. Multilingual part-of-speech tagging: Two unsupervised approaches. *Journal of Artificial Intelligence Research*, 36:341–385.

Sujith Ravi and Kevin Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *Proceedings of ACL-IJCNLP 2009*, pages 504–512, Suntec, Singapore, August.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of EMNLP-CoNLL 2007*, pages 410–420.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: training log-linear models on unlabeled data. In *Proceedings of ACL 2005*, pages 354–362, Morristown, NJ, USA.

K. Toutanova and M. Johnson. 2007. A Bayesian LDA-based model for semi-supervised part-of-speech tagging. In *Proceedings of NIPS 2007*.

Jurgen Van Gael, Andreas Vlachos, and Zoubin Ghahramani. 2009. The infinite HMM for unsupervised PoS tagging. In *Proceedings of EMLNP 2009*, pages 678–687, Singapore, August.

Andreas Vlachos, Anna Korhonen, and Zoubin Ghahramani. 2009. Unsupervised and constrained dirichlet process mixture models for verb clustering. In *Proceedings of GEMS 2009*, pages 74–82, Morristown, NJ, USA.

We're Not in Kansas Anymore: Detecting Domain Changes in Streams

†⋆**Mark Dredze** and †∗**Tim Oates** and †‡**Christine Piatko**
†Human Language Technology Center of Excellence,
⋆Center for Language and Speech Processing,
‡Applied Physics Lab
Johns Hopkins University
∗University of Maryland, Baltimore County
mdredze@cs.jhu.edu, oates@umbc.edu, christine.piatko@jhuapl.edu

Abstract

Domain adaptation, the problem of adapting a natural language processing system trained in one domain to perform well in a different domain, has received significant attention. This paper addresses an important problem for deployed systems that has received little attention – detecting when such adaptation is needed by a system operating in the wild, i.e., performing classification over a stream of unlabeled examples. Our method uses $\mathcal{A}$-distance, a metric for detecting shifts in data streams, combined with classification margins to detect domain shifts. We empirically show effective domain shift detection on a variety of data sets and shift conditions.

1 Introduction

Consider a named entity recognition system trained on newswire stories. Given annotated documents containing sentences like "*Tony Hayward* has faced fresh criticism for taking time off to go sailing ..." we would like to learn a model that will allow us to recognize that "Obama" and "BP" are named entities in a sentence like "Obama summoned BP executives ...". When all of the documents come from one data distribution, like newswire articles, this tends to work well. However, the sentence "OBAMA SUMMONED BP EXECUTIVES ..." from transcribed broadcast news, and others like it, will probably lead to poor results because the features it relies on have changed. For example, capitalization patterns are no longer a good indicator of the presence of a named entity and appositives are not indicated by punctuation. This problem of *domain shift* is a pervasive problem in NLP in which any kind of model – a parser, a POS tagger, a sentiment classifier – is tested on data that do not match the training data.

Given a model and a stream of unlabeled instances, we are interested in automatically *detecting* changes in the feature distribution that negatively impact classification accuracy. For example, a sentiment classification model trained on book reviews may heavily weight n-grams features like "uplifting" and "page turner". Those features may never occur in reviews of kitchen appliances that get mixed in at test time, and useful features in this new domain like "efficient" and "noisy compressor" will have never been seen during training and therefore not be in the model. Furthermore, we do not assume labeled instances are available to help detect these harmful changes. Other tasks related to changes in data distributions, like detecting concept drift in which the labeling function changes, may require labeled instances, but that is not the focus of this paper.

There is significant work on the related problem of adapting a classifier for a known domain shift. Versions of this problem include adapting using only unlabeled target domain data (Blitzer et al., 2006; Blitzer et al., 2007; Jiang and Zhai, 2007), adapting using a limited amount of target domain labeled data (Daumé, 2007; Finkel and Manning, 2009), and learning across multiple domains simultaneously in an online setting (Dredze and Crammer, 2008b). However, in practical settings, we do not know *if* the data distribution will change, and certainly not when. Additionally, we will not know to *what* do-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 585–595,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

main the shift will happen. A discussion forum devoted to science fiction books may change over time to focus more on fantasy and then narrow to discussions of vampire fiction. Maybe this shift is harmless and it is possible to identify the sentiment of the discussants with the original model with no loss in accuracy. If not, we seek methods that detect this shift and trigger the use of an adaptation method.

Our domain shift detection problem can be decomposed into two subproblems: detecting distributional changes in streams of real numbers, and representing a stream of examples as a stream of real numbers informative for distribution change detection. We select the $\mathcal{A}$-distance metric (Kifer et al., 2004) to solve the first subproblem since it has been previously used in other domain adaptation work (Blitzer et al., 2006; Blitzer et al., 2007). Our main contribution is towards the second problem, representing examples as real numbers for this task. We demonstrate that classification margins, which incorporate information about features that most impact system accuracy, can effectively solve the second subproblem. Furthermore, we show that the previously proposed Confidence Weighted learning algorithm (Dredze et al., 2008) can provide a more informative measure than a simple margin for this task. Our experiments include evaluations on commonly used domain adaptation data and false change scenarios, as well as comparisons to *supervised* detection methods that observe label values, or have knowledge of the target domain.

We begin with a description of our task and previous applications to language data. After describing the data used in this paper, we discuss the $\mathcal{A}$-distance metric and how it has previously been used for adaptation. We then show that margin based methods effectively capture information to detect domain shifts, and propose an alternate way of generating informative margin values. Finally, we compare our results to settings with supervised knowledge, and close with a survey of related work.

2 Domain Shifts in Language Data

The study of domain shifts in language data has been the purview of domain adaptation and transfer learning, which seek to adapt or transfer a model learned on one source domain with labeled data to another target domain with few or no labeled examples. Formally, errors from such transfers have two sources: differences in feature distributions and changes to labeling functions (annotation standards) (Ben-David et al., 2006; Ben-David et al., 2009). Empirical work on NLP domain shifts has focused on the former. For example, Blitzer et al. (2007) learned correspondences between features across domains and Jiang and Zhai (2007) weighted source domain examples by their similarity to the target distribution.

We continue in this tradition by making two assumptions about our setting. First, a change in domain will be signaled by a change in the feature distributions. That is, new words, phrases, syntactic structures, etc. signal that the system has shifted to a new domain. Second, while there may be a change in the labeling function, i.e., features have a different meaning in each domain, this will be a secondary concern. For example, both Daumé (2007) and Dredze and Crammer (2008b) assume that domains are more similar than different.

A similar problem to the one we consider is that of concept drift, where a stream of examples are labeled with a shifting labeling function (concept) (Nishida and Yamauchi, 2007; Widmer and Kubat, 1996). While concept drift is similar there are two important differences. First, concept drift can be measured using a stream of *labeled* examples, so system accuracy is directly measured. For example, Klinkenberg and Joachims (2000) detect concept drift with support vector machines, using estimates of leave-one-out performance to adaptively adjust and maintain a training window that minimizes estimated generalization error. This is possible only because class labels arrive with the examples in the stream. Another concept drift detection algorithm, STEPD, uses a statistical test to continually monitor the possibly changing stream, measuring system accuracy directly, again using the labels it receives for each example (Nishida, 2008). Obviously, no such labels are available in our unsupervised setting. Second, concept drift assumes only changes in the labeling function, whereas domain adaptation relies on feature distribution changes.

Several properties of detecting domain shifts in natural language streams distinguish it from traditional domain adaptation, concept drift, and other related tasks:

- **No Target Distribution Examples** Blitzer et al. (2007) estimate the loss in accuracy from domain shift by discriminating between two data distributions. In our setting, we have no knowledge of the target distribution.

- **No Labeled Target Data** Some approaches to domain adaptation assume a limited number of labeled examples (Daumé, 2007; Dredze and Crammer, 2008b; Finkel and Manning, 2009). We assume no labels in our setting.

- **Online Setting** Domain adaptation typically assumes a batch transfer between two domains. We consider a purely stream (online) setting.

- **Computationally Constrained** Our approach must be fast, as we expect to run our domain shift detector alongside a deployed NLP system. This limits both computation and storage.

- **Unknown Adaptation** A critical assumption of previous work is that a domain change has occurred. We must ascertain this ourselves.

Despite these challenges, we show unsupervised stream-based methods that effectively identify shifts in domain in language data. Furthermore, our methods are tied directly to the learning task so are sensitive to changes in actual task accuracy. Our methods have low false positive rates of change detection, which is important since examples within a single domain display a large amount of variance, which could be mistaken for a domain change.

Once a change is detected, any number of actions may be appropriate. The maintainer of the system may be notified that performance is suffering, labels can be obtained for a sample of instances from the stream for retraining, or large volumes of unlabeled instances can be used for instance reweighting (Jiang and Zhai, 2007).

3 Datasets

We begin the presentation of our methods by describing the data used in our experiments. We selected three data sets commonly used in domain adaptation: spam (Jiang and Zhai, 2007), ACE 2005 named entity recognition (Jiang and Zhai, 2007), and sentiment (Blitzer et al., 2007). Sentiment and spam are binary and ACE is multi-class. Note that in all experiments, a shift in the domain yields a decrease in system accuracy.

The goal of the spam data is to classify an email (bag-of-words) as either spam or ham (not-spam). Each email user may have different preferences and features. We used unigram and bigram features, following Dredze and Crammer (2008b) for feature extraction, and used the three task A users as three domains. The ACE 2005 named entity recognition dataset includes 7 named entity class labels (person, organization, location, geopolitical entity, facility, vehicle, weapon) for 5 text genres (newswire, broadcast news, broadcast conversations, conversational telephone speech, weblogs). We use 4000 examples from each genre and used Jiang and Zhai's feature-extracted data.[1] The sentiment data contains reviews from Amazon for four product types: books, dvds, electronics, and kitchen. We include an additional two types (music and video from Dredze and Crammer) in our false shift experiments and use unigram and bigram features, following Blitzer et al.

4 The $\mathcal{A}$-Distance

Our approach to detecting domain shifts in data streams that negatively impact system accuracy is based on the ability to (1) detect distributional changes in streams of real numbers and (2) convert document streams to streams of informative real numbers. This section describes how we achieve the former, and the next section describes the latter.

Theoretical work on domain adaptation showed that the $\mathcal{A}$-distance (Kifer et al., 2004), a stream based measure of difference between two arbitrary probability distributions P and P', can be used to evaluate the difference between two domain distributions (Ben-David et al., 2006). In a batch setting this corresponds to learning a linear classifier to discriminate the domains, and Blitzer et al. (2007) showed correlations with the error from domain adaptation. Given our interest in streaming data we return to the original stream formulation of $\mathcal{A}$-distance.

The $\mathcal{A}$-distance detects differences between two arbitrary probability distributions by dividing the range of a random variable into a set of (possibly

[1] We thank Jing Jiang for the feature-extracted ACE data.

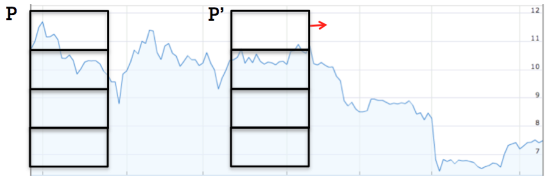


Figure 1: The $\mathcal{A}$-distance is computed between two windows (P and P' in a stream of real-valued data. The samples in each window are divided into intervals, and the $\mathcal{A}$-distance measures the change in the distributions over these intervals between the two windows.

overlapping) intervals, and then measures changes in the probability that a value drawn for that variable falls into any one of the intervals. If such a change is large, a change in the underlying distribution is declared. Let $\mathcal{A}$ be a set of real intervals and let $A \in \mathcal{A}$ be one such interval. For that interval, $P(A)$ is the probability that a value drawn from some unknown distribution falls in A. The $\mathcal{A}$-distance between P and P', i.e. the difference between two distributions over the intervals, is defined as follows:

$$d_{\mathcal{A}}(P, P') = 2 \sup {}_{A \in \mathcal{A}} |P(A) - P'(A)|.$$

Two distributions are said to be different when, for a user-specified threshold ϵ, $d_{\mathcal{A}}(P, P') > \epsilon$. The $\mathcal{A}$-distance is distribution independent. That is, it makes no assumptions about the form of the underlying distribution nor about the form of the change that might occur, either algorithmically or in the underlying theory. Unlike the L_1 norm, the $\mathcal{A}$-distance can be shown to require finitely many samples to detect distribution differences, a property that is crucial for streaming, sample-based approaches.

Since the $\mathcal{A}$-distance processes a stream of real numbers, we need to represent an example using a real number, such as the classification margin for that example. The first n of these numbers in the stream are a sample from P, and the most recent n are a sample from P'. We signal a domain shift when the $\mathcal{A}$-distance between P and P' is large (greater than ϵ). Larger values of n result in more accurate estimates of $P(A)$ and slower detection of changes.

The two windows of samples of size n are shown graphically in Fig. 1. Each increment on the horizontal axis represents the arrival of a new document. The vertical axis is some value computed from each document, such as its classification margin. To compute P and P', one needs to specify $\mathcal{A}$ and n, which are shown as two stacks of boxes that are identical except for their position. The width of each box is n, the number of examples used to estimate $P(A)$ and $P(A')$ for $A \in \mathcal{A}$, where the real interval A corresponds to the vertical span of the box. The value $P(A)$ is simply the number of documents whose real value falls inside that interval A divided by n. Note that the first n documents in the stream are used to compute P, and as each new document arrives the location of the stack of boxes used to compute P' is shifted to the right by one.

In Fig. 1, the number of examples whose real value falls in the top two intervals for P is approximately the same, with no example's value falling in the lower two intervals. For P', almost every one of the n document values falls in the second interval from the top, virtually assuring that $d_{\mathcal{A}}(P, P')$ will be large. Though the intervals in the figure do not overlap, they typically do.

Given n and intervals $\mathcal{A}$, the value of ϵ is chosen by randomization testing. Because the $\mathcal{A}$-distance is distribution independent, a sample of size $m \gg n$ is drawn from any distribution that spans $\mathcal{A}$. This sample is treated as a stream as described above, and the largest value of $d_{\mathcal{A}}(P, P')$ is stored. The sample is permuted and this process is repeated l times. Note that any change detection would be a false positive because all values were sampled from the same distribution. The values $d_{\mathcal{A}}(P, P')$ are sorted from largest to smallest, and ϵ is chosen to be the $\lfloor \alpha l \rfloor^{th}$ such value where parameter α is a user specified false positive probability.

Both the time and space complexity of our approach based on the $\mathcal{A}$-distance are small. Given n and $\mathcal{A}$, n instances must be stored in the sliding window and $2|\mathcal{A}|$ counters are required to represent P and P'. Note that both values are constants based on user specified parameters, not on the size of the stream. Processing a new instance involves computing its margin and updating P and P', all of which can occur in constant time.

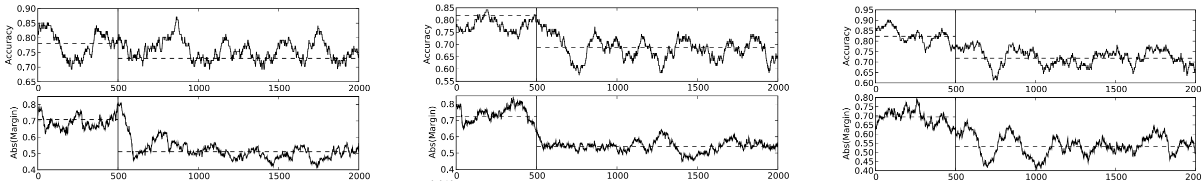


Figure 2: Each column of plots is a representative result using an SVM on a single run over a sentiment data shift: dvds → electronics, electronics → books, and kitchen → books, from left to right. The horizontal axis is the number of instances from the stream processed by the classifier. The top plot is the accuracy of the classifier on the last 100 instances. The bottom plot is the absolute value of the SVM classification margin. The vertical line at 500 instances marks the point of domain shift. Horizontal dotted lines indicate the mean of the accuracy/margin before and after the domain shift. Note that in all cases, the mean accuracy drops, as do the mean margin values, demonstrating that both can indicate domain shifts.

5 $\mathcal{A}$-Distance Over Margins

Since shifts in domains correlate with changes in distributions, it is natural to begin by considering the observed features in each example. When we shift from a source domain (e.g., book reviews) to a target domain (e.g., dvd reviews) we expect a change in the distribution for common source words (“author” and “plot” become less common). Since the $\mathcal{A}$-distance assumes a stream of single values, we can apply an $\mathcal{A}$-distance detector to each feature (e.g., unigram and bigram count) individually. However, our extensive experiments with this approach (omitted here) show that it suffers from a number of flaws, such as a high false positive rate if all features are tracked, the difficult problem of identifying an informative subset of features for tracking, and deciding how many such features need to change before a shift has occurred, which turns out to be highly variable between shifts.

Therefore, our goal is to use a single $\mathcal{A}$-distance tracker by collapsing each example to a single value. One way of doing this is to consider the *classification margin* produced by the classifier. The margin weighs features by their importance in classification. When more important features disappear, we expect the magnitude of the margin to decrease. Additionally, features that change but do not influence system performance are effectively ignored since they do not influence the margin. This approach has the advantage of task sensitivity, only tracking changes that impact task accuracy. Initial experiments showed effectiveness with the unsigned (absolute value of the) margin, which we use in all experiments.

We begin by examining visually the information content of the margin with regards to predicting a domain shift. The caption of Fig. 2 describes the setup, and the first row of the figure illustrates the effects of the shift on the source domain classifier’s empirical accuracy, measured on a window of the previous 100 examples. The horizontal dashed lines indicate the average accuracy before and after the shift. Note that in each case, average classification accuracy drops after the shift. However, at any one point the accuracy displays considerable variance. Thus, while classification accuracy clearly suffers, it is difficult to measure this even in a supervised setting with labeled examples when considering a small portion of the stream.

The second row of Fig. 2 shows the average unsigned margin value of an SVM classifier computed over the previous 100 examples in the stream. The two dashed horizontal lines indicate the average margin value over source and target examples. There is a clear drop in the average margin value after the shift. This difference suggests that the margin can be examined directly to detect a domain shift. However, these values vary considerably so extracting useful information is not trivial.

We evaluated the ability of $\mathcal{A}$-distance trackers to detect such changes in margin values by simulating domain shifts using each domain pair in a task (books to dvds, weblogs to newswire, etc.). For each domain shift setting, we first trained a classifier on 1000 source domain instances. In our experiments, we used three different classification algorithms: Support Vector Machines (SVM) (Chang

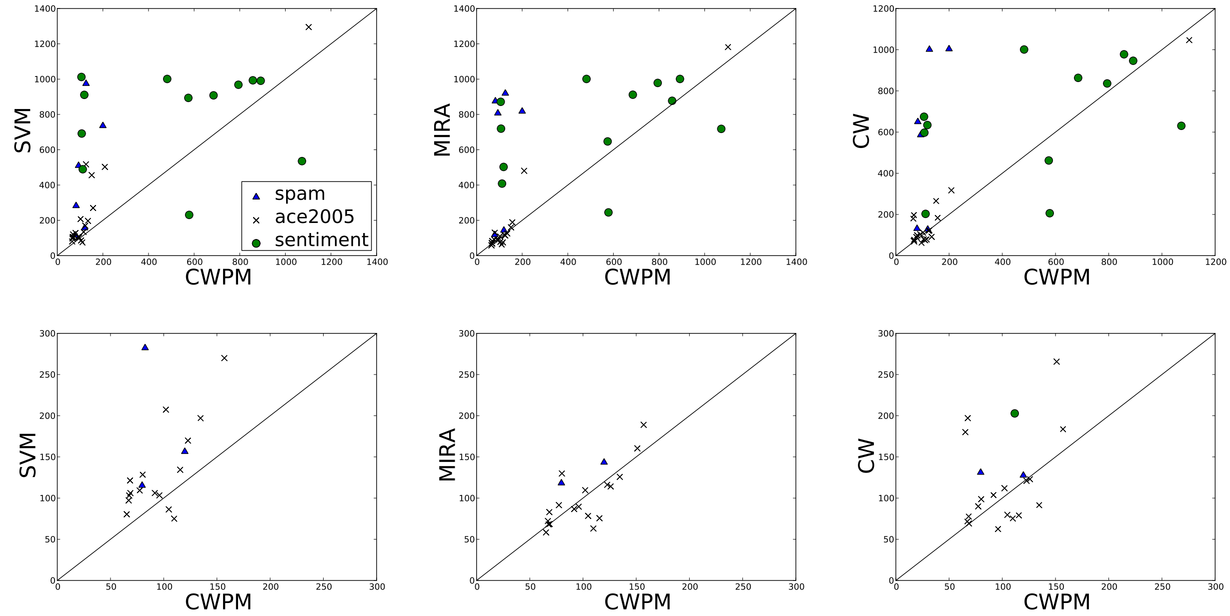


Figure 3: The mean number of instances after a domain change at which the $\mathcal{A}$-distance tracker detects a change. Each point represents the mean number of instances for CWPM (x-axis) and the SVM, MIRA and CW methods (y-axis). Datasets are indicated by different markers. The second row zooms each plot to the bottom left corner of the first row. Points above the diagonal indicate SVM, MIRA or CW took longer to detect a change than CWPM.

and Lin, 2001), MIRA (Crammer et al., 2006) and Confidence Weighted (CW) learning (Dredze et al., 2008). We evaluated each trained classifier on 500 test examples to measure accuracy on the source domain, and then used it to label examples in a stream. The first 500 examples in the stream were used for calibrating our change detection methods. The next 500 examples were from the source domain, followed by 1500 examples from the target domain. Over these 2000 examples we ran each of our detection methods. Experiments were repeated over 10 fixed random data permutations.

We automatically select $\mathcal{A}$-distance intervals as follows. First, we computed the mean and variance of the 500 calibration margins and then added intervals for .5 standard deviations away from the mean in each direction, .5 to 1 standard deviation in each direction, and intervals for 1 standard deviation to $\pm\infty$. We also added three evenly spaced overlapping intervals. To calibrate a FP rate of 0.05 we sampled from a Gaussian with the above mean and variance and used $n = 200$, $m = 10000$ and $l = 50$.

The results for each experiment (38 shifts repeated averaged over 10 runs each) are shown in Fig. 3.[2] Each plot represents one of the three classifiers (SVM, MIRA, CW) plotted on the vertical axis, where each point's y-value indicates the number of examples observed after a shift occurred before the $\mathcal{A}$-distance detector registered a change. Smaller values (lower points) are preferred. The second row of plots highlights the 0 to 300 region of the first row. (The x-axis will be discussed in the next section.) Notice that in many cases, a change was registered within 300 examples, showing that domain shifts can be reasonably detected using the margin values alone.

Equally important to detecting changes is robustness to false changes. We evaluated the margin detector for false positives in two ways. First, we logged any incorrectly detected changes before the shift. For all three algorithms, there were very few false positives (Table 1). The highest false positive rate was about 1% (CW), while for the SVM experiments, not a single detector fired prematurely in any experiment.

[2] The method plotted on the x-axis will be introduced in the next section. To evaluate the three methods in this section (SVM, MIRA, CW) compare the y-values.

Second, we sought to test the robustness of the method over a long stream of examples where no change occurred. In this experiment, we selected 11 domains that had a sufficient number of examples to consider a long stream of source domain examples.[3] Rather than use 500 source domain examples followed by 1500 target domain examples, all 2000 examples were from the source domain. All other settings were the same. For the SVM detector, out of 110 runs we detected 6 false positives, 3 of which were for the same data set (kitchen) (see Table 1.)

6 Confidence Weighted Margins

In the previous section, we showed that margin values could be used to detect domain shifts. We now explore ways to reduce the number of target domain examples needed to detect domain shift by improving the margin values.

Margin values are often taken as a measure of prediction confidence. From this perspective, the $\mathcal{A}$-distance margin tracker identifies when prediction confidence drops. Another task that relies on margins as measures of confidence is active learning, where uncertainty sampling for margin based systems is determined based on the magnitude of the predicted margin. Dredze and Crammer (2008a) showed how Confidence Weighted (CW) learning could be used to generate a more informative measure of confidence for active learning.

CW is an online algorithm inspired by the MIRA update (Crammer et al., 2006), which ensures a positive margin while minimizing parameter change. CW replaces the Euclidean distance used in the MIRA update with the KL divergence over Gaussian distributions. CW learning maintains a Gaussian distribution over linear weight vectors with mean $\boldsymbol{\mu} \in \mathbb{R}^N$ and diagonal covariance $\Sigma \in \mathbb{R}^{N \times N}$.

Maintaining a distribution over prediction functions is appropriate for our task where we consider margin values as confidence. We replace the margin $|\boldsymbol{w} \cdot \boldsymbol{x}|$, where $\boldsymbol{w}$ is a standard linear classifier, with a probabilistic margin $|\left(\Pr_{\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)} \left[\text{sign}(\boldsymbol{w} \cdot \boldsymbol{z}) = 1\right]\right) - \frac{1}{2}|$. Dredze and Crammer showed that this probabilistic margin can be translated into a *corrected* geometric margin, which is computed as the normalized margin as $\bar{\text{M}} = M/\sqrt{V}$, where M is the mean $M = \boldsymbol{\mu} \cdot \boldsymbol{x}$ and V the variance $V = \boldsymbol{x}^\top \Sigma \boldsymbol{x}$ of a univariate Gaussian distribution over the unsigned-margin $\text{M} = \boldsymbol{w} \cdot \boldsymbol{x}$. We call this method CWPM, for Confidence Weighted Probabilistic Margin.

We compared using CWPM to the standard margins produced by an SVM, MIRA and CW classifier in the last section. Fig. 3 shows the results of these comparisons. In each plot, CWPM (normalized margin) is plotted on the x-axis, indicating how many examples from the target domain were observed before the detector identified a change. The y-axis in each plot is the number of instances observed for the SVM, MIRA and CW methods. As before, each point is the average of the 10 randomized runs used above (assuming that detectors that did not fire do so at the end of the stream.) Points above the diagonal indicate that CWPM detected a change sooner than the comparative method. Of the 38 shifts, CWPM detected domain shifts faster than an SVM 34 times, MIRA 26 times and CW 27 times.

We repeated the experiments to detect false positives for each margin based method. Table 1 shows the false positives for the 38 domain shifts considered as well as the 11 false shift domain shifts. The false positive rates are among the lowest for CWPM. This shows that CWPM is a more useful indicator for detecting domain changes.

7 Gradual Shifts

We have shown detection of sudden shifts between the source and target domains. However, some shifts may happen gradually over time. We evaluate this by modifying the stream as follows: the first 500 instances come from the source domain, and the remaining 1500 are sampled randomly from the source and target domains. The probability of an instance being drawn from the target domain at time i is $p_i(\boldsymbol{x} = \text{target}) = \frac{i}{1500}$, where i counts from the start of the shift at index 500. The probability of sampling target domain data increases uniformly over the stream. At index 750 after the start of the shift each domain is equally likely. The ACE and Sentiment datasets had sufficient data to be evaluated in this setting. Fig. 4 shows CWPM still performs best, but results are close (SVM: 22 of 32, MIRA &

[3] ACE: bc, bn, cts, nw, wl; Sentiment: books, dvd, electronics, kitchen, music, video

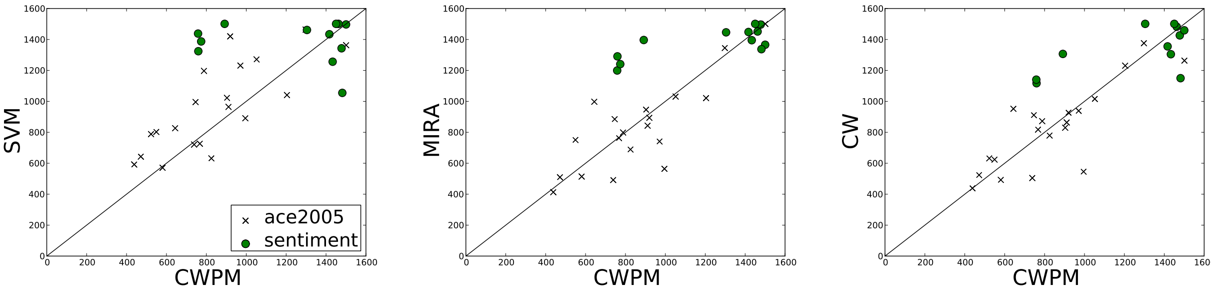


Figure 4: Gradual shift detection with SVM, MIRA or CW vs. CWPM. There were no false positives.

	Domain Shift FPs	
Algorithm	*True Shift*	*False Shift*
Sec. 5: SVM	0	6
Sec. 6: MIRA	2	13
Sec. 6: CW	5	10
Sec. 6: CWPM	1	6
Total tests	380	110

Table 1: False positives (FPs) observed in true domain shift and false domain shift experiments for methods in corresponding sections. Each setting was run 10 times, resulting in 380 true domain shifts and 110 false shifts.

CW: 17 of 32). As expected, detections happen later in the stream. The closer results are likely due to the increased difficulty of the task. With less clear information, there it is more difficult for all the algorithms to recognize a change, and performance across the methods begins to equalize. Even in this more difficult setting, CWPM is the best performer.

8 Comparison to Supervised Information

So far we have considered applying $\mathcal{A}$-distance tracking to information freely available in a real world system: the classification margins. As a useful baseline for comparison, we can measure using *supervised* sources of information, where additional information is provided that is not normally available. In particular, we investigate two types of supervised knowledge: the labels of examples in the stream and knowledge of the target domain. In each case, we compare using the $\mathcal{A}$-distance and CWPM versus applying the $\mathcal{A}$-distance to supervised information.

8.1 Classifier Accuracy

In Sec. 4 we showed that both the margin and recent classifier accuracy indicate when shifts in domains occur (Fig. 2). We developed techniques based on the margin, which is available at test time. We now consider knowledge of the true labels for these test examples, which allows for tracking classifier accuracy. We can use the $\mathcal{A}$-distance to detect when unexpected changes in accuracy occur.

For each test example classified by the system, we evaluated whether the system was correct in its prediction by examining the label. If the classifier was correct, we output a 1; otherwise, we output a 0. Over this 1/0 stream produced by checking classifier accuracy we ran an $\mathcal{A}$-distance detector, with intervals set for 1s and 0s (10,000 uniform samples to calibrate the threshold for a false positive rate of 0.05.) If an unusual number of 0s or 1s occur – more or less mistakes than on the source domain – a change is detected.[4] Results on this accuracy stream are compared to CWPM (Fig. 5.) Despite this supervised information, CWPM still detects domain changes faster than with labeled examples. Consider again Fig. 2, which shows both accuracy and margin values over time. While the average accuracy drops, the instantaneous value is very noisy, suggesting that even this additional information may not yield better domain shift detection. This will be interesting to explore in future work.

[4] An alternate approach would be to measure accuracy directly as a real valued number. However, our experiments showed the discrete approach to be more effective.

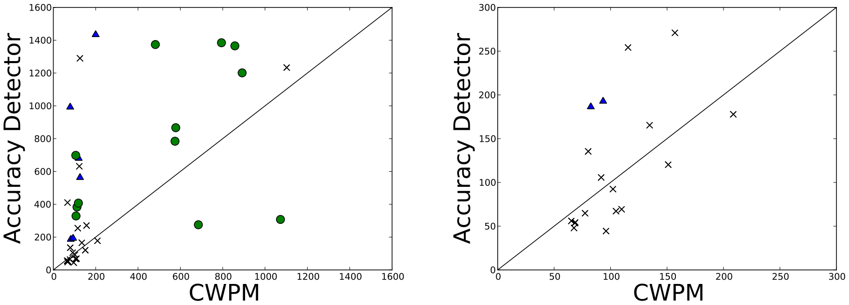


Figure 5: An $\mathcal{A}$-distance accuracy detector, run over a stream of 1s and 0s indicating correct and incorrect predictions of the classifier on examples in a stream. The bulk of points above the line indicate that CWPM is more effective at detecting domain change. CWPM had a single false positive and the accuracy detector had no false positives.

8.2 Domain Classification

Next, we consider another source of supervision: a selection of examples known to be from the target domain. In this setting, we know that a shift will occur and we know to which domain it will occur. This requires a sample of (unlabeled) target domain examples when the target domain is not known ahead of time. Using a common approach to detecting domain differences when data is available from both domains (Ben-David et al., 2009; Blitzer et al., 2007; Rai et al., 2010), we train a binary classifier to differentiate between the source and target domain. We learn a CW classifier on 1000 examples (500 from each domain) that do not appear in the test stream. We then label each example as either "source" or "target" and output a 1 or 0 accordingly. Over this 0/1 stream, we run an $\mathcal{A}$-distance detector with two intervals, one for 1s and one for 0s. The remaining setup is identical to the $\mathcal{A}$-distance experiments above.

Fig. 6 shows the detection rate of CWPM versus $\mathcal{A}$-distance over the domain classifier stream. As expected, the detection rate for the domain classifier is very fast, in almost every case (save 1) less than 400 examples after the shift happens. When CWPM is slow to detect a change (over 400 examples), the domain classifier is the clear winner. However, in the majority of experiments, especially for ACE and spam data, both detectors register a change quickly. These results suggest that while a sample of target domain examples is very helpful, our CWPM approach can also be effective when such samples are not available.

9 Related Work

Early NLP work in the *unsupervised* setting monitored classification confidence values, setting a confidence threshold based on a break-even heuristic, monitoring the rate of (presumed) irrelevant examples based on this threshold, and signaling a change when this rate increased (Lanquillon, 1999).

Confidence estimation has been used for specific NLP components such as information extraction. The correctness of fields extracted via a conditional random field extractor has been shown to correlate well to an estimate obtained by a constrained forward-backward technique (Culotta and McCallum, 2004). EM-based confidence estimation has been used to estimate the confidence of patterns derived from partially supervised relation extraction (Agichtein, 2006). Confidence estimation has also been used to improve the overall effectiveness of NLP systems. Confidence estimates obtained via neural networks have shown gains for speech recognition, spoken language understanding, and machine translation (Gandrabur et al., 2006). Pipeline models using confidence estimates at one stage as weights for further downstream stages improve over baseline dependency parsing and named entity recognition pipeline models (Bunescu, 2008).

An alternative formulation of domain adaptation trains on different corpora from many different domains, then uses linear combinations of models trained on the different corpora(McClosky et al., 2010).

Work in novelty detection is relevant to the task of detecting domain shifts (Scholkopf et al., 2000),

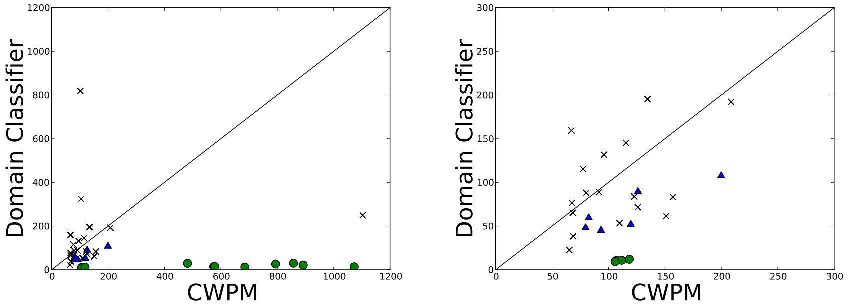


Figure 6: $\mathcal{A}$-distance over a stream of 1s and 0s produced by a supervised classifier trained to differentiate between the source and target domain. Samples from the unseen target domain is very effective. However, for many shifts, the margin based $\mathcal{A}$-distance detector is still competitive. CWPM had a single false positive while the domain classifier stream had 2 false positives in these experiments.

though the rate of occurrence of novel instances is more informative in our setting than the mere fact that novel instances are observed.

We are also motivated by the problem of detecting *genre shift* in addition to domain shift, as in the ACE 2005 data set shifts from newswire to transcripts and blogs. Different text genres occur in traditional settings, such as broadcast news transcripts and newswire, and have begun to proliferate with the variety of social media technologies now available including weblogs. Static genre classification has been explored using a variety of techniques, including exploiting punctuation (Kessler et al., 1997; Dewdney et al., 2001), TF-IDF statistics (Lee and Myaeng, 2002), and part-of-speech statistics and histograms (Finn and Kushmerick, 2006; Feldman et al., 2009).

Finally, statistical estimation in a streaming context has been considered in data mining applications (Muthukrishnan, 2005). Change detection via sequential hypothesis testing has been effective for streaming applications such as network intrusion detection (Muthukrishnan et al., 2007). Detecting new events in a stream of Twitter posts can be done using constant time and space similarity measures based on a modification of locality sensitive hashing (Petrović et al., 2010).

10 Conclusion

While there are a number of methods for domain adaptation, a system first needs to determine that a domain shift has occurred. We have presented methods for automatically detecting such domain shifts from a stream of (unlabeled) examples that require limited computation and memory by virtue of operating on fixed-size windows of data. Our methods were evaluated empirically on a variety of domain shifts using NLP data sets and are shown to be sensitive to shifts while maintaining a low rate of false positives. Additionally, we showed improved detection results using a probabilistic margin based on Confidence Weighted learning. Comparisons to detection with supervised information show that our results are effective even in unlabeled settings. Our methods are promising as tools to accompany the deployment of domain adaptation algorithms, so that a complete system can first identify when a domain shift has occurred before automatically adapting to the new domain.

Acknowledgments

Thanks to the HLTCOE text processing group for many helpful discussions.

References

Eugene Agichtein. 2006. Confidence estimation methods for partially supervised information extraction. In *SDM*.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2006. Analysis of representations for domain adaptation. In *NIPS*.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Vaughan. 2009. A theory of learning from different domains. *Machine Learning*.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Association for Computational Linguistics (ACL)*.

Razvan C. Bunescu. 2008. Learning with probabilistic features for improved pipeline models. In *EMNLP*.

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research (JMLR)*.

Aron Culotta and Andrew McCallum. 2004. Confidence estimation for information extraction. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*.

Hal Daumé. 2007. Frustratingly easy domain adaptation. In *Association for Computational Linguistics (ACL)*.

Nigel Dewdney, Carol VanEss-Dykema, and Richard MacMillan. 2001. The form is the substance: classification of genres in text. In *Workshop on Human Language Technology and Knowledge Management*.

Mark Dredze and Koby Crammer. 2008a. Active learning with confidence. In *Association for Computational Linguistics (ACL)*.

Mark Dredze and Koby Crammer. 2008b. Online methods for multi-domain learning and adaptation. In *EMNLP*.

Mark Dredze, Koby Crammer, and Fernando Pereira. 2008. Confidence-weighted linear classification. In *ICML*.

S. Feldman, M. A. Marin, M. Ostendorf, and M. R. Gupta. 2009. Part-of-speech histograms for genre classification of text. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Jenny Rose Finkel and Christopher D. Manning. 2009. Hierarchical Bayesian domain adaptation. In *NAACL-HLT*.

Aidan Finn and Nicholas Kushmerick. 2006. Learning to classify documents according to genre: Special topic section on computational analysis of style. *J. Am. Soc. Inf. Sci. Technol.*, 57(11):1506–1518.

Simona Gandrabur, George Foster, and Guy Lapalme. 2006. Confidence estimation for NLP applications. *ACM Trans. Speech Lang. Process.*, 3(3):1–29.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in NLP. In *Association for Computational Linguistics (ACL)*.

Brett Kessler, Geoffrey Numberg, and Hinrich Schütze. 1997. Automatic detection of text genre. In *Association for Computational Linguistics (ACL)*.

Daniel Kifer, Shai Ben-David, and Johannes Gehrke. 2004. Detecting change in data streams. In *Very Large Data Bases (VLDB)*.

Ralf Klinkenberg and Thorsten Joachims. 2000. Detecting concept drift with support vector machines. In *International Conference on Machine Learning (ICML)*.

C. Lanquillon. 1999. Information filtering in changing domains. In *IJCAI*.

Yong-Bae Lee and Sung Hyon Myaeng. 2002. Text genre classification with genre-revealing and subject-revealing features. In *SIGIR*.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *NAACL-HLT*, pages 28–36, Los Angeles, California, June. Association for Computational Linguistics.

S. Muthukrishnan, Eric van den Berg, and Yihua Wu. 2007. Sequential change detection on data streams. In *IEEE International Conference on Data Mining Workshops (ICDMW)*.

S. Muthukrishnan. 2005. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 1(2).

Kyosuke Nishida and Koichiro Yamauchi. 2007. Detecting concept drift using statistical testing. In *Discovery Science*.

Kyosuke Nishida. 2008. *Learning and Detecting Concept Drift*. Ph.D. thesis, Hokkaido University, Japan.

Saša Petrović, Miles Osborne, and Victor Lavrenko. 2010. Streaming first story detection with application to twitter. In *NAACL-HLT*, pages 181–189, June.

P. Rai, A. Saha, H. Daumé III, and S. Venkatasubramanian. 2010. Domain Adaptation meets Active Learning. In *Workshop on Active Learning for Natural Language Processing (ALNLP)*, page 27.

Bernhard Scholkopf, Robert Williamson, Alex Smola, John Shawe-Taylor, and John Platt. 2000. Support vector method for novelty detection. In *NIPS*.

Gerhard Widmer and Miroslav Kubat. 1996. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23:69–101.

A Fast Fertility Hidden Markov Model for Word Alignment Using MCMC

Shaojun Zhao and **Daniel Gildea**
Department of Computer Science
University of Rochester

Abstract

A word in one language can be translated to zero, one, or several words in other languages. Using word fertility features has been shown to be useful in building word alignment models for statistical machine translation. We built a fertility hidden Markov model by adding fertility to the hidden Markov model. This model not only achieves lower alignment error rate than the hidden Markov model, but also runs faster. It is similar in some ways to IBM Model 4, but is much easier to understand. We use Gibbs sampling for parameter estimation, which is more principled than the neighborhood method used in IBM Model 4.

1 Introduction

IBM models and the hidden Markov model (HMM) for word alignment are the most influential statistical word alignment models (Brown et al., 1993; Vogel et al., 1996; Och and Ney, 2003). There are three kinds of important information for word alignment models: **lexicality**, **locality** and **fertility**. IBM Model 1 uses only lexical information; IBM Model 2 and the hidden Markov model take advantage of both lexical and locality information; IBM Models 4 and 5 use all three kinds of information, and they remain the state of the art despite the fact that they were developed almost two decades ago.

Recent experiments on large datasets have shown that the performance of the hidden Markov model is very close to IBM Model 4. Nevertheless, we believe that IBM Model 4 is essentially a better model because it exploits the fertility of words in the target language. However, IBM Model 4 is so complex that most researches use the GIZA++ software package (Och and Ney, 2003), and IBM Model 4 itself is treated as a black box. The complexity in IBM Model 4 makes it hard to understand and to improve. Our goal is to build a model that includes lexicality, locality, and fertility; and, at the same time, to make it easy to understand. We also want it to be accurate and computationally efficient.

There have been many years of research on word alignment. Our work is different from others in essential ways. Most other researchers take either the HMM alignments (Liang et al., 2006) or IBM Model 4 alignments (Cherry and Lin, 2003) as input and perform post-processing, whereas our model is a potential replacement for the HMM and IBM Model 4. Directly modeling fertility makes our model fundamentally different from others. Most models have limited ability to model fertility. Liang et al. (2006) learn the alignment in both translation directions jointly, essentially pushing the fertility towards 1. ITG models (Wu, 1997) assume the fertility to be either zero or one. It can model phrases, but the phrase has to be contiguous. There have been works that try to simulate fertility using the hidden Markov model (Toutanova et al., 2002; Deng and Byrne, 2005), but we prefer to model fertility directly.

Our model is a coherent generative model that combines the HMM and IBM Model 4. It is easier to understand than IBM Model 4 (see Section 3). Our model also removes several undesired properties in IBM Model 4. We use Gibbs sampling instead of a heuristic-based neighborhood method for parameter

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 596–605,
MIT, Massachusetts, USA, 9-11 October 2010.

estimation. Our distortion parameters are similar to IBM Model 2 and the HMM, while IBM Model 4 uses inverse distortion (Brown et al., 1993). Our model assumes that fertility follows a Poisson distribution, while IBM Model 4 assumes a multinomial distribution, and has to learn a much larger number of parameters, which makes it slower and less reliable. Our model is much faster than IBM Model 4. In fact, we will show that it is also faster than the HMM, and has lower alignment error rate than the HMM.

Parameter estimation for word alignment models that model fertility is more difficult than for models without fertility. Brown et al. (1993) and Och and Ney (2003) first compute the Viterbi alignments for simpler models, then consider only some neighbors of the Viterbi alignments for modeling fertility. If the optimal alignment is not in those neighbors, this method will not be able find the optimal alignment. We use the Markov Chain Monte Carlo (MCMC) method for training and decoding, which has nice probabilistic guarantees. DeNero et al. (2008) applied the Markov Chain Monte Carlo method to word alignment for machine translation; they do not model word fertility.

2 Statistical Word Alignment Models

2.1 Alignment and Fertility

Given a source sentence $\mathbf{f}_1^J = f_1, f_2, \ldots, f_J$ and a target sentence $\mathbf{e}_1^I = e_1, e_2, \ldots, e_I$, we define the alignments between the two sentences as a subset of the Cartesian product of the word positions. Following Brown et al. (1993), we assume that each source word is aligned to exactly one target word. We denote as $\mathbf{a}_1^J = a_1, a_2, \ldots, a_J$ the alignments between $\mathbf{f}_1^J$ and $\mathbf{e}_1^I$. When a word f_j is not aligned with any word e, a_j is 0. For convenience, we add an empty word ϵ to the target sentence at position 0 (i.e., $e_0 = \epsilon$). However, as we will see, we have to add more than one empty word for the HMM. In order to compute the "jump probability" in the HMM model, we need to know the position of the aligned target word for the previous source word. If the previous source word aligns to an empty word, we could use the position of the empty word to indicate the nearest previous source word that does not align to an empty word. For this reason, we use a total of $I + 1$ empty words for the HMM model[1]. Moore (2004) also suggested adding multiple empty words to the target sentence for IBM Model 1. After we add $I + 1$ empty words to the target sentence, the alignment is a mapping from source to target word positions:

$$a : j \rightarrow i, i = a_j$$

where $j = 1, 2, \ldots, J$ and $i = 1, 2, \ldots, 2I + 1$. Words from position $I + 1$ to $2I + 1$ in the target sentence are all empty words.

We allow each source word to align with exactly one target word, but each target word may align with multiple source words.

The fertility ϕ_i of a word e_i at position i is defined as the number of aligned source words:

$$\phi_i = \sum_{j=1}^{J} \delta(a_j, i)$$

where δ is the Kronecker delta function:

$$\delta(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$$

In particular, the fertility of all empty words in the target sentence is $\sum_{i=I+1}^{2I+1} \phi_i$. We define $\phi_\epsilon \equiv \sum_{i=I+1}^{2I+1} \phi_i$. For a bilingual sentence pair $\mathbf{e}_1^{2I+1}$ and $\mathbf{f}_1^J$, we have $\sum_{i=1}^{I} \phi_i + \phi_\epsilon = J$.

The inverted alignments for position i in the target sentence are a set B_i, such that each element in B_i is aligned with i, and all alignments of i are in B_i. Inverted alignments are explicitly used in IBM Models 3, 4 and 5, but not in our model, which is one reason that our model is easier to understand.

2.2 IBM Model 1 and HMM

IBM Model 1 and the HMM are both generative models, and both start by defining the probability of alignments and source sentence given the target sentence: $P(\mathbf{a}_1^J, \mathbf{f}_1^J | \mathbf{e}_1^{2I+1})$; the data likelihood can be computed by summing over alignments:

[1] If f_{j-1} does not align with an empty word and f_j aligns with an empty word, we want to record the position of the target word that f_{j-1} aligns with. There are $I + 1$ possibilities: f_j is the first word in the source sentence, or f_{j-1} aligns with one of the target word.

$P(\mathbf{f}_1^J|\mathbf{e}_1^{2I+1}) = \sum_{\mathbf{a}_1^J} P(\mathbf{a}_1^J, \mathbf{f}_1^J|\mathbf{e}_1^{2I+1})$. The alignments $\mathbf{a}_1^J$ are the hidden variables. The expectation maximization algorithm is used to learn the parameters such that the data likelihood is maximized.

Without loss of generality, $P(\mathbf{a}_1^J, \mathbf{f}_1^J|\mathbf{e}_1^{2I+1})$ can be decomposed into **length probabilities**, **distortion probabilities** (also called alignment probabilities), and **lexical probabilities** (also called translation probabilities):

$$\begin{aligned} & P(\mathbf{a}_1^J, \mathbf{f}_1^J|\mathbf{e}_1^{2I+1}) \\ & = P(J|\mathbf{e}_1^{2I+1}) \prod_{j=1}^{J} P(a_j, f_j|f_1^{j-1}, a_1^{j-1}, \mathbf{e}_1^{2I+1}) \\ & = P(J|\mathbf{e}_1^{2I+1}) \prod_{j=1}^{J} \Big(P(a_j|f_1^{j-1}, a_1^{j-1}, \mathbf{e}_1^{2I+1}) \times \\ & \qquad P(f_j|f_1^{j-1}, a_1^{j}, \mathbf{e}_1^{2I+1}) \Big) \end{aligned}$$

where $P(J|\mathbf{e}_1^{2I+1})$ is a length probability, $P(a_j|f_1^{j-1}, a_1^{j-1}, \mathbf{e}_1^{2I+1})$ is a distortion probability and $P(f_j|f_1^{j-1}, a_1^{j}, \mathbf{e}_1^{2I+1})$ is a lexical probability.

IBM Model 1 assumes a uniform distortion probability, a length probability that depends only on the length of the target sentence, and a lexical probability that depends only on the aligned target word:

$$P(\mathbf{a}_1^J, \mathbf{f}_1^J|\mathbf{e}_1^{2I+1}) = \frac{P(J|I)}{(2I+1)^J} \prod_{j=1}^{J} P(f_j|e_{a_j})$$

The hidden Markov model assumes a length probability that depends only on the length of the target sentence, a distortion probability that depends only on the previous alignment and the length of the target sentence, and a lexical probability that depends only on the aligned target word:

$$P(\mathbf{a}_1^J, \mathbf{f}_1^J|\mathbf{e}_1^{2I+1}) = \\ P(J|I) \prod_{j=1}^{J} P(a_j|a_{j-1}, I) P(f_j|e_{a_j})$$

In order to make the HMM work correctly, we enforce the following constraints (Och and Ney, 2003):

$$\begin{aligned} P(i+I+1|i', I) &= p_0 \delta(i, i') \\ P(i+I+1|i'+I+1, I) &= p_0 \delta(i, i') \\ P(i|i'+I+1, I) &= P(i|i', I) \end{aligned}$$

where the first two equations imply that the probability of jumping to an empty word is either 0 or p_0, and the third equation implies that the probability of jumping from a non-empty word is the same as the probability of jumping from the corespondent empty word.

The absolute position in the HMM is not important, because we re-parametrize the distortion probability in terms of the distance between adjacent alignment points (Vogel et al., 1996; Och and Ney, 2003):

$$P(i|i', I) = \frac{c(i - i')}{\sum_{i''} c(i'' - i')}$$

where $c(\,)$ is the count of jumps of a given distance.

In IBM Model 1, the word order does not matter. The HMM is more likely to align a source word to a target word that is adjacent to the previous aligned target word, which is more suitable than IBM Model 1 because adjacent words tend to form phrases.

For these two models, in theory, the fertility for a target word can be as large as the length of the source sentence. In practice, the fertility for a target word in IBM Model 1 is not very big except for rare target words, which can become a garbage collector, and align to many source words (Brown et al., 1993; Och and Ney, 2003; Moore, 2004). The HMM is less likely to have this garbage collector problem because of the alignment probability constraint. However, fertility is an inherent cross-language property and these two models cannot assign consistent fertility to words. This is our motivation for adding fertility to these two models, and we expect that the resulting models will perform better than the baseline models. Because the HMM performs much better than IBM Model 1, we expect that the fertility hidden Markov model will perform much better than the fertility IBM Model 1. Throughout the paper, “our model” refers to the fertility hidden Markov model.

Due to space constraints, we are unable to provide details for IBM Models 3, 4 and 5; see Brown et al. (1993) and Och and Ney (2003). But we want to point out that the locality property modeled in the HMM is missing in IBM Model 3, and is modeled invertedly in IBM Model 4. IBM Model 5 removes deficiency (Brown et al., 1993; Och and Ney, 2003) from IBM Model 4, but it is computationally very expensive due to the larger number of parameters than IBM Model 4, and IBM Model 5 often provides no improvement on alignment accuracy.

3 Fertility Hidden Markov Model

Our fertility IBM Model 1 and fertility HMM are both generative models and start by defining the probability of fertilities (for each non-empty target word and all empty words), alignments, and the source sentence given the target sentence: $P(\phi_1^I, \phi_\epsilon, \mathbf{a}_1^J, \mathbf{f}_1^J | \mathbf{e}_1^{2I+1})$; the data likelihood can be computed by summing over fertilities and alignments: $P(\mathbf{f}_1^J | \mathbf{e}_1^{2I+1}) = \sum_{\phi_1^I, \phi_\epsilon, \mathbf{a}_1^J} P(\phi_1^I, \phi_\epsilon, \mathbf{a}_1^J, \mathbf{f}_1^J | \mathbf{e}_1^{2I+1})$.

The fertility for a non-empty word e_i is a random variable ϕ_i, and we assume ϕ_i follows a Poisson distribution $\mathrm{Poisson}(\phi_i; \lambda(e_i))$. The sum of the fertilities of all the empty words (ϕ_ϵ) grows with the length of the target sentence. Therefore, we assume that ϕ_ϵ follows a Poisson distribution with parameter $I\lambda(\epsilon)$.

Now $P(\phi_1^I, \phi_\epsilon, \mathbf{a}_1^J, \mathbf{f}_1^J | \mathbf{e}_1^{2I+1})$ can be decomposed in the following way:

$$\begin{aligned}
&P(\phi_1^I, \phi_\epsilon, \mathbf{a}_1^J, \mathbf{f}_1^J | \mathbf{e}_1^{2I+1}) \\
&\quad = P(\phi_1^I | \mathbf{e}_1^{2I+1}) P(\phi_\epsilon | \phi_1^I, \mathbf{e}_1^{2I+1}) \times \\
&\qquad \prod_{j=1}^{J} P(a_j, f_j | f_1^{j-1}, a_1^{j-1}, \mathbf{e}_1^{2I+1}, \phi_1^I, \phi_\epsilon) \\
&\quad = \prod_{i=1}^{I} \frac{\lambda(e_i)^{\phi_i} e^{-\lambda(e_i)}}{\phi_i!} \times \\
&\qquad \frac{(I\lambda(\epsilon))^{\phi_\epsilon} e^{-I\lambda(\epsilon)}}{\phi_\epsilon!} \times \\
&\qquad \prod_{j=1}^{J} \Big(P(a_j | f_1^{j-1}, a_1^{j-1}, \mathbf{e}_1^{2I+1}, \phi_1^I, \phi_\epsilon) \times \\
&\qquad\quad P(f_j | f_1^{j-1}, a_1^{j}, \mathbf{e}_1^{2I+1}, \phi_1^I, \phi_\epsilon) \Big)
\end{aligned}$$

Superficially, we only try to model the length probability more accurately. However, we also enforce the fertility for the same target word across the corpus to be consistent. The expected fertility for a non-empty word e_i is $\lambda(e_i)$, and the expected fertility for all empty words is $I\lambda(\epsilon)$. Any fertility value has a non-zero probability, but fertility values that are further away from the mean have low probability. IBM Models 3, 4, and 5 use a multinomial distribution for fertility, which has a much larger number of parameters to learn. Our model has only one parameter for each target word, which can be learned more reliably.

In the fertility IBM Model 1, we assume that the distortion probability is uniform, and the lexical probability depends only on the aligned target word:

$$\begin{aligned}
&P(\phi_1^I, \phi_\epsilon, \mathbf{a}_1^J, \mathbf{f}_1^J | \mathbf{e}_1^{2I+1}) \\
&\quad = \prod_{i=1}^{I} \frac{\lambda(e_i)^{\phi_i} e^{-\lambda(e_i)}}{\phi_i!} \times \\
&\qquad \frac{(I\lambda(\epsilon))^{\phi_\epsilon} e^{-(I\lambda(\epsilon))}}{\phi_\epsilon!} \times \\
&\qquad \frac{1}{(2I+1)^J} \prod_{j=1}^{J} P(f_j | e_{a_j}) \qquad (1)
\end{aligned}$$

In the fertility HMM, we assume that the distortion probability depends only on the previous alignment and the length of the target sentence, and that the lexical probability depends only on the aligned target word:

$$\begin{aligned}
&P(\phi_1^I, \phi_\epsilon, \mathbf{a}_1^J, \mathbf{f}_1^J | \mathbf{e}_1^{2I+1}) \\
&\quad = \prod_{i=1}^{I} \frac{\lambda(e_i)^{\phi_i} e^{-\lambda(e_i)}}{\phi_i!} \times \\
&\qquad \frac{(I\lambda(\epsilon))^{\phi_\epsilon} e^{-(I\lambda(\epsilon))}}{\phi_\epsilon!} \times \\
&\qquad \prod_{j=1}^{J} P(a_j | a_{j-1}, I) P(f_j | e_{a_j}) \qquad (2)
\end{aligned}$$

When we compute $P(\mathbf{f}_1^J | \mathbf{e}_1^{2I+1})$, we only sum over fertilities that agree with the alignments:

$$P(\mathbf{f}_1^J | \mathbf{e}_1^{2I+1}) \quad = \quad \sum_{\mathbf{a}_1^J} P(\mathbf{a}_1^J, \mathbf{f}_1^J | \mathbf{e}_1^{2I+1})$$

where

$$\begin{aligned} & P(\mathbf{a}_1^J, \mathbf{f}_1^J | \mathbf{e}_1^{2I+1}) \\ &= \sum_{\phi_1^I, \phi_\epsilon} P(\phi_1^I, \phi_\epsilon, \mathbf{a}_1^J, \mathbf{f}_1^J | \mathbf{e}_1^{2I+1}) \\ &\approx P(\phi_1^I, \phi_\epsilon, \mathbf{a}_1^J, \mathbf{f}_1^J | \mathbf{e}_1^{2I+1}) \times \\ &\quad \prod_{i=1}^{I} \delta\left(\sum_{j=1}^{J} \delta(a_j, i), \phi_i\right) \times \\ &\quad \delta\left(\sum_{i=I+1}^{2I+1} \sum_{j=1}^{J} \delta(a_j, i), \phi_\epsilon\right) \qquad (3) \end{aligned}$$

In the last two lines of Equation 3, ϕ_ϵ and each ϕ_i are not free variables, but are determined by the alignments. Because we only sum over fertilities that are consistent with the alignments, we have $\sum_{\mathbf{f}_1^J} P(\mathbf{f}_1^J | \mathbf{e}_1^{2I+1}) < 1$, and our model is deficient, similar to IBM Models 3 and 4 (Brown et al., 1993). We can remove the deficiency for fertility IBM Model 1 by assuming a different distortion probability: the distortion probability is 0 if fertility is not consistent with alignments, and uniform otherwise. The total number of consistent fertility and alignments is $\frac{J!}{\phi_\epsilon! \prod_{j=1}^{J} \phi_i!}$. Replacing $\frac{1}{(2I+1)^J}$ with $\frac{\phi_\epsilon! \prod_{j=1}^{J} \phi_i!}{J!}$, we have:

$$\begin{aligned} & P(\phi_1^I, \phi_\epsilon, \mathbf{a}_1^J, \mathbf{f}_1^J | \mathbf{e}_1^{2I+1}) \\ &= \prod_{i=1}^{I} \lambda(e_i)^{\phi_i} e^{-\lambda(e_i)} \times \\ &\quad (I\lambda(\epsilon))^{\phi_\epsilon} e^{-(I\lambda(\epsilon))} \times \\ &\quad \frac{1}{J!} \prod_{j=1}^{J} P(f_j | e_{a_j}) \end{aligned}$$

In our experiments, we did not find a noticeable change in terms of alignment accuracy by removing the deficiency.

4 Expectation Maximization Algorithm

We estimate the parameters by maximizing $P(\mathbf{f}_1^J | \mathbf{e}_1^{2I+1})$ using the expectation maximization (EM) algorithm (Dempster et al., 1977). The auxiliary function is:

$$\begin{aligned} & L(P(f|e), P(a|a'), \lambda(e), \xi_1(e), \xi_2(a')) \\ &= \sum_{\mathbf{a}_1^J} \tilde{P}(\mathbf{a}_1^J | \mathbf{e}_1^{2I+1}, \mathbf{f}_1^J) \log P(\mathbf{a}_1^J, \mathbf{f}_1^J | \mathbf{e}_1^{2I+1}) \\ &\quad - \sum_{e} \xi_1(e)(\sum_{f} P(f|e) - 1) \\ &\quad - \sum_{a'} \xi_2(a')(\sum_{a} P(a|a') - 1) \end{aligned}$$

Because $P(\mathbf{a}_1^J, \mathbf{f}_1^J | \mathbf{e}_1^{2I+1})$ is in the exponential family, we get a closed form for the parameters from expected counts:

$$P(f|e) = \frac{\sum_s c(f|e; \mathbf{f}^{(s)}, \mathbf{e}^{(s)})}{\sum_f \sum_s c(f|e; \mathbf{f}^{(s)}, \mathbf{e}^{(s)})} \qquad (4)$$

$$P(a|a') = \frac{\sum_s c(a|a'; \mathbf{f}^{(s)}, \mathbf{e}^{(s)})}{\sum_a \sum_s c(a|a'; \mathbf{f}^{(s)}, \mathbf{e}^{(s)})} \qquad (5)$$

$$\lambda(e) = \frac{\sum_s c(\phi|e; \mathbf{f}^{(s)}, \mathbf{e}^{(s)})}{\sum_s c(k|e; \mathbf{f}^{(s)}, \mathbf{e}^{(s)})} \qquad (6)$$

where s is the number of bilingual sentences, and

$$\begin{aligned} c(f|e; \mathbf{f}_1^J, \mathbf{e}_1^{2I+1}) &= \sum_{\mathbf{a}_1^J} \tilde{P}(\mathbf{a}_1^J | \mathbf{f}_1^J, \mathbf{e}_1^{2I+1}) \times \\ &\quad \sum_j \delta(f_j, f) \delta(e_i, e) \\ c(a|a'; \mathbf{f}_1^J, \mathbf{e}_1^{2I+1}) &= \sum_{\mathbf{a}_1^J} \tilde{P}(\mathbf{a}_1^J | \mathbf{f}_1^J, \mathbf{e}_1^{2I+1}) \times \\ &\quad \sum_j \delta(a_j, a) \delta(a_{j-1}, a') \\ c(\phi|e; \mathbf{f}_1^J, \mathbf{e}_1^{2I+1}) &= \sum_{\mathbf{a}_1^J} \tilde{P}(\mathbf{a}_1^J | \mathbf{f}_1^J, \mathbf{e}_1^{2I+1}) \times \\ &\quad \sum_i \phi_i \delta(e_i, e) \\ c(k|e; \mathbf{f}_1^J, \mathbf{e}_1^{2I+1}) &= \sum_i k(e_i) \delta(e_i, e) \end{aligned}$$

These equations are for the fertility hidden Markov model. For the fertility IBM Model 1, we do not need to estimate the distortion probability.

5 Gibbs Sampling for Fertility HMM

Although we can estimate the parameters by using the EM algorithm, in order to compute the expected counts, we have to sum over all possible alignments $\mathbf{a}_1^J$, which is, unfortunately, exponential. We developed a Gibbs sampling algorithm (Geman and Geman, 1984) to compute the expected counts.

For each target sentence $\mathbf{e}_1^{2I+1}$ and source sentence $\mathbf{f}_1^J$, we initialize the alignment a_j for each source word f_j using the Viterbi alignments from IBM Model 1. During the training stage, we try all $2I + 1$ possible alignments for a_j but fix all other alignments.[2] We choose alignment a_j with probability $P(a_j|a_1, \cdots a_{j-1}, a_{j+1} \cdots a_J, \mathbf{f}_1^J, \mathbf{e}_1^{2I+1})$, which can be computed in the following way:

$$P(a_j|a_1, \cdots, a_{j-1}, a_{j+1}, \cdots, a_J, \mathbf{f}_1^J, \mathbf{e}_1^{2I+1}) = \frac{P(\mathbf{a}_1^J, \mathbf{f}_1^J|\mathbf{e}_1^{2I+1})}{\sum_{a_j} P(\mathbf{a}_1^J, \mathbf{f}_1^J|\mathbf{e}_1^{2I+1})} \quad (7)$$

For each alignment variable a_j, we choose t samples. We scan through the corpus many times until we are satisfied with the parameters we learned using Equations 4, 5, and 6. This Gibbs sampling method updates parameters constantly, so it is an “online learning” algorithm. However, this sampling method needs a large amount of communication between machines in order to keep the parameters up to date if we compute the expected counts in parallel. Instead, we do “batch learning”: we fix the parameters, scan through the entire corpus and compute expected counts in parallel (E-step); then combine all the counts together and update the parameters (M-step). This is analogous to what IBM models and the HMM do in the EM algorithms. The algorithm for the E-step on one machine (all machines are independent) is in Algorithm 1.

For the fertility hidden Markov model, updating $P(\mathbf{a}_1^J, \mathbf{f}_1^J|\mathbf{e}_1^{2I+1})$ whenever we change the alignment a_j can be done in constant time, so the complexity of choosing t samples for all a_j $(j = 1, 2, \ldots, J)$ is $O(tIJ)$. This is the same complexity as the HMM if t is $O(I)$, and it has lower complexity if t is a constant. Surprisingly, we can achieve better results than the HMM by computing as few as 1 sample for each alignment, so the fertility hidden Markov model is much faster than the HMM. Even when choosing t such that our model is 5 times faster than the HMM, we achieve better results.

Algorithm 1: One iteration of E-step: draw t samples for each a_j for each sentence pair $(\mathbf{f}_1^J, \mathbf{e}_1^{2I+1})$ in the corpus

```
for $(\mathbf{f}_1^J, \mathbf{e}_1^{2I+1})$ in the corpus do
    Initialize $\mathbf{a}_1^J$ with IBM Model 1;
    for $t$ do
        for $j$ do
            for $i$ do
                $a_j = i$;
                Compute $P(\mathbf{a}_1^J, \mathbf{f}_1^J|\mathbf{e}_1^{2I+1})$;
            end
            Draw a sample for $a_j$ using
            Equation 7;
            Update counts;
        end
    end
end
```

We also consider initializing the alignments using the HMM Viterbi algorithm in the E-step. In this case, the fertility hidden Markov model is not faster than the HMM. Fortunately, initializing using IBM Model 1 Viterbi does not decrease the accuracy in any noticeable way, and reduces the complexity of the Gibbs sampling algorithm.

In the testing stage, the sampling algorithm is the same as above except that we keep the alignments $\mathbf{a}_1^J$ that maximize $P(\mathbf{a}_1^J, \mathbf{f}_1^J|\mathbf{e}_1^{2I+1})$. We need more samples in the testing stage because it is unlikely to get to the optimal alignments by sampling a few times for each alignment. On the contrary, in the above training stage, although the samples are not accurate enough to represent the distribution defined by Equation 7 for each alignment a_j, it is accurate enough for computing the expected counts, which are defined at the corpus level. Interestingly, we found that throwing away the fertility and using the HMM Viterbi decoding achieves same results as the sampling approach (we can ignore the difference because it is tiny), but is faster. Therefore, we use Gibbs sampling for learning and the HMM Viterbi decoder for testing.

Gibbs sampling for the fertility IBM Model 1 is similar but simpler. We omit the details here.

[2] For fertility IBM Model 1, we only need to compute $I + 1$ values because e_{I+1}^{2I+1} are identical empty words.

Alignment	Model	P	R	AER
en → cn	IBM1	49.6	55.3	47.8
	IBM1F	55.4	57.1	43.8
	HMM	62.6	59.5	39.0
	HMMF-1	65.4	59.1	37.9
	HMMF-5	66.8	60.8	36.2
	HMMF-30	67.8	62.3	34.9
	IBM4	66.8	64.1	34.5
cn → en	IBM1	52.6	53.7	46.9
	IBM1F	55.9	56.4	43.9
	HMM	66.1	62.1	35.9
	HMMF-1	68.6	60.2	35.7
	HMMF-5	71.1	62.2	33.5
	HMMF-30	71.1	62.7	33.2
	IBM4	69.3	68.5	31.1

Table 1: AER results. IBM1F refers to the fertility IBM1 and HMMF refers to the fertility HMM. We choose $t = 1$, 5, and 30 for the fertility HMM.

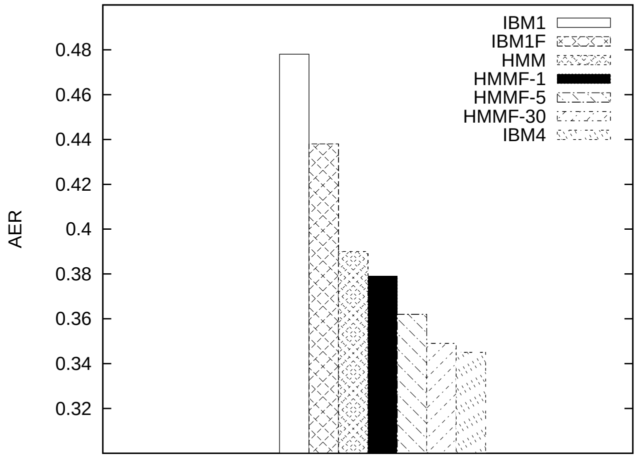


Figure 1: AER comparison (en→cn)

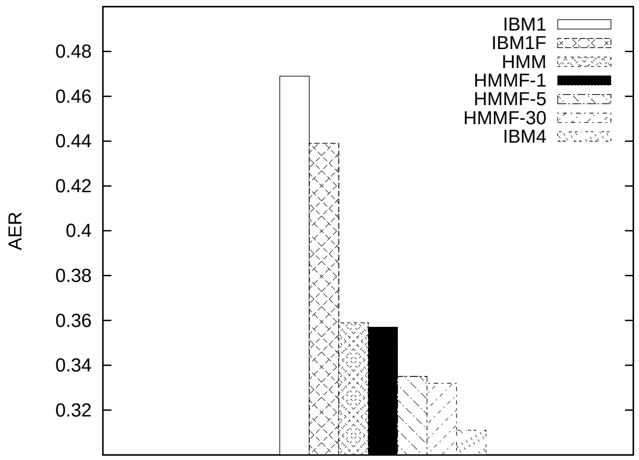


Figure 2: AER comparison (cn →en)

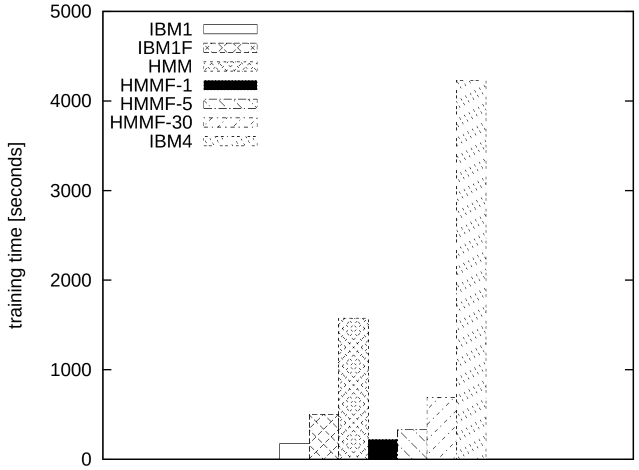


Figure 3: Training time comparison. The training time for each model is calculated from scratch. For example, the training time of IBM Model 4 includes the training time of IBM Model 1, the HMM, and IBM Model 3.

6 Experiments

We evaluated our model by computing the word alignment and machine translation quality. We use the alignment error rate (AER) as the word alignment evaluation criterion. Let A be the alignments output by word alignment system, P be a set of possible alignments, and S be a set of sure alignments both labeled by human beings. S is a subset of P.

Precision, recall, and AER are defined as follows:

$$\begin{aligned} \text{recall} &= \frac{|A \cap S|}{|S|} \\ \text{precision} &= \frac{|A \cap P|}{|A|} \\ \text{AER}(S, P, A) &= 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|} \end{aligned}$$

AER is an extension to F-score. Lower AER is better.

We evaluate our fertility models on a Chinese-English corpus. The Chinese-English data taken from FBIS newswire data, and has 380K sentence pairs, and we use the first 100K sentence pairs as our training data. We used hand-aligned data as reference. The Chinese-English data has 491 sentence pairs.

We initialize IBM Model 1 and the fertility IBM Model 1 with a uniform distribution. We smooth all parameters ($\lambda(e)$ and $P(f|e)$) by adding a small value (10^{-8}), so they never become too small. We run both models for 5 iterations. AER results are computed using the IBM Model 1 Viterbi alignments, and the Viterbi alignments obtained from the Gibbs sampling algorithm.

We initialize the HMM and the fertility HMM with the parameters learned in the 5th iteration of IBM Model 1. We smooth all parameters ($\lambda(e)$, $P(a|a')$ and $P(f|e)$) by adding a small value (10^{-8}). We run both models for 5 iterations. AER results are computed using traditional HMM Viterbi decoding for both models.

It is always difficult to determine how many samples are enough for sampling algorithms. However, both fertility models achieve better results than their baseline models using a small amount of samples. For the fertility IBM Model 1, we sample 10 times for each a_j, and restart 3 times in the training stage; we sample 100 times and restart 12 times in the testing stage. For the fertility HMM, we sample 30 times for each a_j with no restarting in the training stage; no sampling in the testing stage because we use traditional HMM Viterbi decoding for testing. More samples give no further improvement.

Initially, the fertility IBM Model 1 and fertility HMM did not perform well. If a target word e only appeared a few times in the training corpus, our model cannot reliably estimate the parameter $\lambda(e)$. Hence, smoothing is needed. One may try to solve it by forcing all these words to share a same parameter $\lambda(e_{\text{infrequent}})$. Unfortunately, this does not solve the problem because all infrequent words tend to have larger fertility than they should. We solve the problem in the following way: estimate the parameter $\lambda(e_{\text{non\_empty}})$ for all non-empty words, all infrequent words share this parameter. We consider words that appear less than 10 times as infrequent words.

Table 1, Figure 1, and Figure 2 shows the AER results for different models. We can see that the fertility IBM Model 1 consistently outperforms IBM Model 1, and the fertility HMM consistently outperforms the HMM.

The fertility HMM not only has lower AER than the HMM, it also runs faster than the HMM. Figure 3 show the training time for different models. In fact, with just 1 sample for each alignment, our model archives lower AER than the HMM, and runs more than 5 times faster than the HMM. It is possible to use sampling instead of dynamic programming in the HMM to reduce the training time with no decrease in AER (often an increase). We conclude that the fertility HMM not only has better AER results, but also runs faster than the hidden Markov model.

We also evaluate our model by computing the machine translation BLEU score (Papineni et al., 2002) using the Moses system (Koehn et al., 2007). The training data is the same as the above word alignment evaluation bitexts, with alignments for each model symmetrized using the grow-diag-final heuristic. Our test is 633 sentences of up to length 50, with four references. Results are shown in Table 2; we see that better word alignment results do not lead to better translations.

Model	BLEU
HMM	19.55
HMMF-30	19.26
IBM4	18.77

Table 2: BLEU results

7 Conclusion

We developed a fertility hidden Markov model that runs faster and has lower AER than the HMM. Our model is thus much faster than IBM Model 4. Our model is also easier to understand than IBM Model 4. The Markov Chain Monte Carlo method used in our model is more principled than the heuristic-based neighborhood method in IBM Model 4. While better word alignment results do not necessarily correspond to better translation quality, our translation results are comparable in translation quality to both the HMM and IBM Model 4.

Acknowledgments We would like to thank Tagyoung Chung, Matt Post, and the anonymous reviewers for helpful comments. This work was supported by NSF grants IIS-0546554 and IIS-0910611.

References

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

Colin Cherry and Dekang Lin. 2003. A probability model to improve word alignment. In *Proceedings of ACL-03*.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the *EM* algorithm. *Journal of the Royal Statistical Society*, 39(1):1–21.

John DeNero, Alexandre Bouchard-Cote, and Dan Klein. 2008. Sampling alignment structure under a Bayesian translation model. In *Proceedings of EMNLP*.

Yonggang Deng and William Byrne. 2005. HMM word and phrase alignment for statistical machine translation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 169–176, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Stuart Geman and Donald Geman. 1984. Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, November.

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.

P. Liang, B. Taskar, and D. Klein. 2006. Alignment by agreement. In *North American Association for Computational Linguistics (NAACL)*, pages 104–111.

Robert C. Moore. 2004. Improving IBM word alignment Model 1. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 518–525, Barcelona, Spain, July.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of ACL-02*.

Kristina Toutanova, H. Tolga Ilhan, and Christopher D. Manning. 2002. Extensions to HMM-based statistical word alignment models. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 87–94.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *COLING-96*, pages 836–841.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Minimum Error Rate Training by Sampling the Translation Lattice

Samidh Chatterje[*]
Department of Computer Science
Florida State University, USA
chatterj@cs.fsu.edu

Nicola Cancedda
Xerox Research Centre Europe
6 Chemin de Maupertuis, 38240 Meylan, France
nicola.cancedda@xrce.xerox.com

Abstract

Minimum Error Rate Training is the algorithm for log-linear model parameter training most used in state-of-the-art Statistical Machine Translation systems. In its original formulation, the algorithm uses N-best lists output by the decoder to grow the *Translation Pool* that shapes the surface on which the actual optimization is performed. Recent work has been done to extend the algorithm to use the entire translation lattice built by the decoder, instead of N-best lists. We propose here a third, intermediate way, consisting in growing the translation pool using samples randomly drawn from the translation lattice. We empirically measure a systematic improvement in the BLEU scores compared to training using N-best lists, without suffering the increase in computational complexity associated with operating with the whole lattice.

1 Introduction

Most state-of-the-art Statistical Machine Translation (SMT) systems are based on a log-linear model of the conditional probability of generating a certain translation given a specific source sentence. More specifically, the conditional probability of a translation **e** and a word alignment **a** given a source sentence **f** is modeled as:

$$P(\mathbf{e},\mathbf{a}|\mathbf{f}) \propto \exp\left(\sum_{k=1}^{K} \lambda_k h_k\left(\mathbf{e},\mathbf{a},\mathbf{f}\right)\right) \qquad (1)$$

where the $h_k(\mathbf{e},\mathbf{a},\mathbf{f})$ are *feature functions* providing complementary sources of information on the quality of the produced translation (and alignment). Once such a model is known: the *decoder* (i.e. the actual translation program), which builds a translation by searching in the space of all possible translations the one that maximizes the conditional probability:

$$(\mathbf{e}^*,\mathbf{a}^*) = \arg\max_{\mathbf{e},\mathbf{a}} \sum_{k=1}^{K} \lambda_k h_K(\mathbf{e},\mathbf{a},\mathbf{f}) \qquad (2)$$

where we have taken into account that the exponential is monotonic.

The parameters λ_k determine the relative importance of the different feature functions in the global score. Best results are typically obtained by searching in the space of all possible parameter vectors $\bar{\lambda}$ for the one that minimizes the error on a held-out development dataset for which one or more reference human translations are available, as measured by some automatic measure. This procedure is referred to as *Minimum Error Rate Training (MERT)*.

1.1 Minimum Error Rate Training on N-best Lists

The most widespread MERT algorithm is the one described in (Och, 2003). This algorithm starts by initializing the parameter vector $\bar{\lambda}$. For each source sentence in the development set, the decoder

[*] The work behind this paper was done during an internship at the Xerox Research Centre Europe. The author was partially supported by NSF through Grant CCF-0643593 and the AFOSR Young Investigator Research Program.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 606–615,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

is used to initialize a translation pool with a list of N-best scoring candidate translations according to the model. Using this pool and the corresponding reference translations then, an optimization procedure is run to update the parameter vector to a $\bar{\lambda}'$ with reduced error. The decoder is then invoked again, the new output N-best list is merged into the translation pool, and the procedure is iterated. The algorithm stops either after a predefined number of iterations or upon convergence, which is reached when no new element is added to the translation pool of any sentence, or when the size of the update in the parameter vector is below a threshold.

The error measure minimized at each iteration is usually BLEU (Papineni et al., 2002). BLEU essentially measures the precision with which the translation produced by a system recovers n-grams of different orders from the available reference translation(s), used as a gold standard.

The optimization procedure that is run within each iteration on the growing translation pools is based on the key observation that BLEU only depends on the single translation receiving the highest score by the translation model (which would be the one shown to the receipient) in the translation pool. This in turn means that, for any given sentence, its contribution to BLEU changes only when the value of the parameters change in such a way that the sentence ranking first according to the model switches from one to another. This situation does not change when one considers all the sentences in a development set instead of just one: while varying the $\bar{\lambda}$ vector, the BLEU score changes only when there is a change at the top of the ranking of the alternatives for at least one sentence in the set. In other words, BLEU is piece-wise constant in $\bar{\lambda}$. MERT then proceeds by performing an iterative line search by fixing each time the value of all components of $\bar{\lambda}$ but one[1]: for such a free parameter a global optimum can be identified by enumerating all the points that cause a change in BLEU. The value of the component is then fixed at the middle of an interval with maximum BLEU, and the procedure is iterated until convergence. Since the error function is highly irregular, and the iterative line search is not guaranteed to converge to a global optimum, the procedure is repeated many times with different initializations, and the best convergence point is retained.

The MERT algorithm suffers from the following problem: it assumes at each iteration that the set of candidates with a chance to make it to the top (for some value of the parameter vector) is well represented in the translation pool. If the translation pool is formed in the standard way by merging N-best lists, this assumption is easily violated in practice. Indeed, the N-best list often contains only candidates displaying minor differences, and represents only a very small sample of alternative possible translations, strongly biased by the current parameter setting.

Recognizing this shortcoming, Macherey et al. (2008) extended the MERT algorithm so as to use the whole set of candidate translations compactly represented in the search lattice produced by the decoder, instead of only a N-best list of candidates extracted from it. This is achieved via an elegant but relatively heavy dynamic programming algorithm that propagates sufficient statistics (called *envelopes*) throughout the whole search graph. The reported theoretical worst-case complexity of this algorithm is $O(|V||\mathcal{E}|\log|\mathcal{E}|)$, where V and $\mathcal{E}$ are the vertex set and the edge set of the lattice respectively.

We propose here an alternative method consisting in sampling a list of candidate translations from the probability distribution induced by the translation lattice. This simple method produces a list of candidates more representative of the complete distribution than an N-best list, side-stepping the intricacies of propagating envelopes throughout the lattice. Computational complexity increases only marginally over the N-best list approach, while still yielding significant improvements in final translation quality.

1.2 The translation lattice

Finding the optimal translation according to Equation 1 is NP-complete (Knight, 1999). Most phrase-based SMT systems resort then to beam-search heuristic algorithms for solving the problem approximately. In their most widespread version, PBSMT decoders proceed by progressively extending translation prefixes by adding one new phrase at a time, and correspondingly "consuming" portions of the

[1] More generally, one can select each time a combination of coordinates identifying a line in the parameter space, and is not restricted to a coordinate direction.

source sentence. Each prefix is associated with a node in a graph, and receives a score according to the model. Whenever two prefixes having exactly the same possible extensions are detected, the lower-scoring one is *merged* into the other, thus creating a re-entrancy in the directed graph, which has then the characteristics of a *lattice* (Figure 1). Edges in the lattice are labelled with the phrase-pair that was used to perform the corresponding extension, the source word positions that were covered in doing the extension, and the corresponding increment in model score.

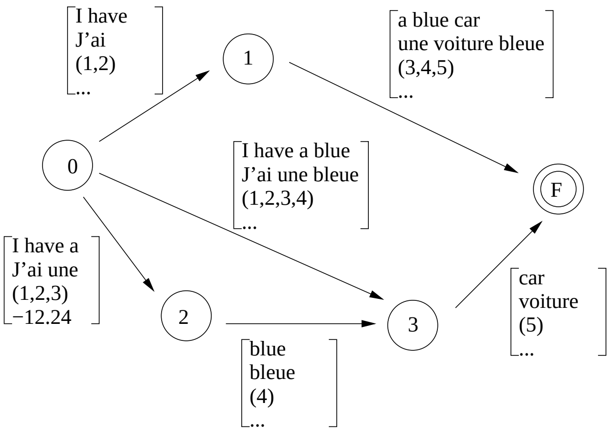


Figure 1: A lattice showing some possible translations of the English sentence: *I have a blue car*. The state with ID 0 is the start state and the one with F is the final state.

2 Related Work

Since its introduction, (Och, 2003) there has been various suggestions for optimizing the MERT criterion. Zens et al. (2007) use the MERT criterion to optimize the N-best lists using the Downhill Simplex Algorithm (Press, 2007). But the Downhill Simplex Algorithm loses its robustness as the dimension goes up by more than 10 (Macherey et al., 2008). Deterministic Annealing was suggested by Smith and Eisner (2006) where the authors propose to minimize the *expected loss* or *risk*. They define the expectation using a probability distribution over hypotheses that they gradually anneal to focus on the 1-best hypothesis. Different search strategies were investigated by Cer et al. (2008). Work has been done to investigate a perceptron-like online margin training for statisitical machine translation (Watanabe et al., 2007). Building on this paper, the most recent work to our knowledge has been done by Chiang et al. (2008). They explore the use of the Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003; Crammer et al., 2006) algorithm instead of MERT. Macherey et al. (2008) propose a new variation of MERT where the algorithm is tuned to work on the whole phrase lattice instead of N-best list only. The new algorithm constructs the error surface of all translations that are encoded in the phrase lattice. They report significant convergence improvements and BLEU score gains over N-best MERT when trained on NIST 2008 translation tasks. More recently, this algorithm was extended to work with hypergraphs encoding a huge number of translations produced by MT systems based on Synchronous Context Free Grammars (Kumar et al., 2009). All the methods cited here work on either N-best lists or from whole translation lattices built by the decoder. To our knowledge, none of them proposes sampling translations from the lattice.

3 Sampling candidate translations from the lattice

In this section we first start by providing an intuition of why we believe it is a good idea to sample from the translation lattice, and then describe in detail how we do it.

3.1 An intuitive explanation

The limited scope of n-best lists rules out many alternative translations that would receive the highest score for some values of the parameter vector. The complete set of translations that can be produced using a fixed phrase table (also called *reachable* translations) for a given source sentence can be represented as a set of vectors in the space spanned by the feature functions (Fig. 2). Not all such translations stand a chance to receive the highest score for any value of the parameter vector, though. Indeed, if translations **h**, **h'**and **h"** are such that $h_k \leq h'_k \leq h''_k$ for all feature k, then there is no value of $\bar{\lambda}$ that will give to **h'** a score higher than both **h** and **h"**. The candidates that would rank first for some value of the $\bar{\lambda}$ parameter vector are those on the convex envelope of the overall candidate set. We know of no effective way to generate this convex envelope in polynomial time. The set of candidates represented by the decoder lattice is a subset (enclosed in the larger dashed polygon in the figure) of this set. This subset is biased to contain translations ranking high according to the values of the parameter vector (the direction labelled with λ) used to produce it, because of the pruning strategies that guide the construction of the translation lattice. Both the N-best list and our proposed random sample are further subsets of the set of translations encoded in the lattice. The N-best list is very biased towards translations that score high with the current choice of parameters: its convex envelope (the smaller dashed polygon) is very different from the one of the complete set of translations, and also from that of the translations in the lattice. The convex envelope of a random sample from the translation lattice (the dotted polygon in the figure), will generally be somewhat closer to the envelope of the whole lattice itself.

The curves in the figure indicate regions of constant loss (e.g. iso-BLEU score, much more irregularly shaped in reality than in the drawing). For this sentence, then, the optimal choice of the parameters would be around λ^*. Performing an optimization step based on the random sample envelope would result in a more marked update (λ'_{sample}) in the direction of the best parameter vector than if an N-best list is used ($\lambda'_{\text{N-best}}$).

Notice that Figure 2 portraits a situation with only two features, for obvious reasons. In practice the number of features will be substantially larger, with values between five and twenty being common practice. In real cases, then, a substantially larger fraction of reachable translations will tend to lie on the convex envelope of the set, and not inside the convex hull.

3.2 The sampling procedure

We propose to modify the standard MERT algorithm and sample N candidates from the translation lattice according to the probability distribution over paths induced by the model, given the current setting of the $\bar{\lambda}$ parameters, instead of using an N-best list. The sampling procedes from the root node of the lattice, corresponding to an empty translation candidate covering no words of the source, by chosing step by step the next edge to follow. The probability distribution for each possible follow-up is the posterior probability of following the edge given the path prefix derived from the lattice: it is obtained via a preliminary backward sweep.

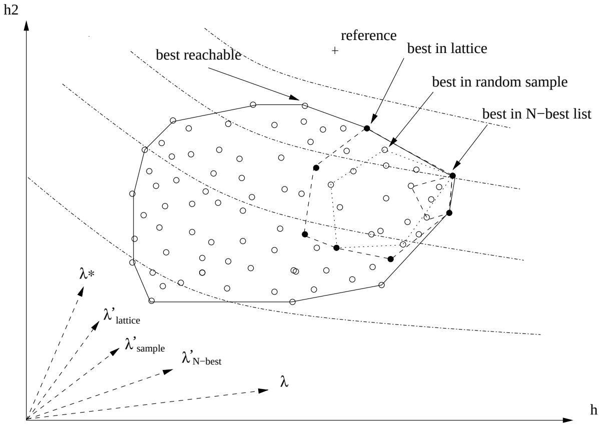


Figure 2: Envelope of the set of reachable translations where the model has two feature functions h_1 and h_2. The envelope of the lattice is the outer dashed polygon, while the envelope of the N-best list is the inner one. Using the whole lattice as translation pool will result in a more marked update towards the optimal parameters. The random sample from the lattice is enclosed by the dotted line. If we use it, we can intuitively expect updates towards the optimum of intermediate effectiveness between those of the N-best list method and those of the lattice method.

Since feature functions are incremental over the edges by design, the non-normalized probability of a path is given by:

$$P(e_1, \ldots, e_m) = e^{\sum_{i=1}^{m} \sigma(e_i)} \qquad (3)$$

where

$$\sigma(e_i) = \sum_{k=1}^{K} \lambda_k h_k(e_i) \qquad (4)$$

is the *score* of edge e_i. With a small abuse of notation we will also denote it as $\sigma(n_{j,k})$, where it is intended that e_i goes from node n_j to node n_k. Let's denote with $\sigma(n_i)$ the score of node n_i, i.e. the logarithm of the cumulative unnormalized probability of all the paths in the lattice that go from node n_i to a final node. The unnormalized probability of selecting node n_j starting from n_i can then be expressed recursively as follows:

$$S(n_j|n_i) \approx e^{(\sigma(n_j)+\sigma(n_{i,j}))} \quad (5)$$

The scores required to compute this sampling probabilities can be obtained by a simple backward pass in the lattice. Let P_i be the set of successors of n_i. So the total unnormalized log-probability of reaching a final state (i.e. with a complete translation) from n_i is given by the equation below.

$$\sigma(n_i) = \log(\sum_{n_j \in P_i} e^{(\sigma(n_j)+\sigma(n_{i,j}))}) \quad (6)$$

where we set $\sigma(n_i) = 0$ if $P_i = \emptyset$, that is if n_i is a final node. At the end of the backward sweep, $\sigma(n_0)$ contains the unnormalized cumulative probability of all paths, i.e. the partition function. Notice that this normalising constant cancels out when computing local sampling probabilities for traversed nodes in the lattice.

Once we know the transition probability (Eq. 5) for each node, we sample by starting in the root node of the lattice and at each step randomly selecting among its successors, until we end in the final node. The whole sampling procedure is repeated as many times as the number of samples sought. After collecting samples for each sentence, the whole list is used to grow the translation pool.

Notice that when using this sampling method it is no longer possible to use the stability of the translation pool as a stopping criterion. The MERT algorithm must thus be run either for a fixed number of iterations, or until the norm of the update to the parameter vector goes below a threshold.

3.3 Time Complexity Analysis

For each line search in the inner loop of the MERT algorithm, all methods considered here need to compute the projection of the convex envelope that can be scanned by leaving all components unchanged but one[2]. If we use either N-best lists or random samples to form the translation pool, and M is the size of the translation pool, then computing the envelope can be done in time $O(M \log M)$ using the *SweepLine* algorithm reproduced as Algorithm 1 in (Macherey et al., 2008). As shown in the same article, the lattice method for computing the envelope is $O(|V||\mathcal{E}| \log |\mathcal{E}|)$, where V is the vertex set of the lattice, and $\mathcal{E}$ is its edge set. In standard decoders there is a maximum limit D to the allowed distortion, and lattice vertices are organized in J priority queues [3] of size at most a, where J is the length of the source sentence and a is a parameter of the decoder set by the user. Also, there is a limit K to the maximum number of source words spanned by a phrase, and only up to c alternative translations for a same source phrase are kept in the phrase table. Under these standard conditions, the number of outgoing edges E' from each lattice vertex can be bounded by a constant. A way to see this is by considering that if an hypothesis is extended with a phrase, then the extended hypothesis must end up in a stack at most K stacks to the right of the original one. There are only aK places in these stacks, so it must be $|E'| \leq aK$.

Since the number of edges leaving each node is bounded by a constant, it is $|E| = \Theta(|V|)$, and the lattice method is $O(|V|^2 \log(|V|))$. The maximum number of vertices in the lattice is limited by the capacity of the stacks: $|V| \leq aJ$. This eventually leads to a complexity of $O(J^2 \log J)$ for the inner loop of the lattice method.

It is interesting to observe that the complexity is driven by the length of the source sentence in the case of the lattice method, and by the size of the translation pool in the case of both the N-best list method and the random sampling method. The latter two methods are asymptotically more effective as long as the size of the sample/N-best list grows subquadratically in the length of the sentence. In most of our experiments we keep the size of the sample constant, independent of the length of the sentence, but other choices can be considered. Since the number of reachable translations grows with the length of the source sentence, length-independent samples explore a smaller fraction of the reachable space. Generating samples (or n-best lists) of size increasing with the length of the source sentence could thus lead to more homogeneous sampling, and possibly a better use of CPU time.

We have so far compared methods in term of the complexity of the innermost loop: the search for a global optimum along a line in the parameter space.

[2] In general, moving along a 1-dimensional subspace of the parameter space.

[3] Traditionally referred to as *stacks*.

This is indeed the most important analysis, since the line search is repeated many times. In order to complete the analysis, we also compare the different methods in terms of the operations that need be performed as part of the outer iteration, that is upon redecoding the development set with a new parameter vector.

The N-best list method requires simply constructing an N-best list from the lattice. This can be done in time linear in the size J of the sentence and in N with a backward sweep in the lattice.

The sampling method requires sampling N times the lattice according to the probability distribution induced by the weights on its edges. We use a dynamic programming approach for computing the posterior probabilities of traversing edges. In this phase we visit each edge of the lattice exactly once, hence this phase is linear in the number of edges in the lattice, hence under the standard assumptions above in the length J of the sentence. Once posterior probabilities are computed for the lattice, we need to sample N paths from it, each of which is composed of at most J edges[4]. Under standard assumptions, randomly selecting the next edge to follow at each lattice node can be done in constant time, so the whole sampling is also $O(NJ)$, like extracting the N-best list.

No operation at all is required by the lattice method in the outer loop, since the whole lattice is passed over for envelope propagation to the inner loop.

4 Experimental Results

Experiments were conducted on the Europarl corpus with the split used for the WMT-08 shared task (Europarl training and test condition) for the language pairs English-French (En-Fr), English-Spanish (En-Es) and English-German (En-De), each in both directions. Training corpora contain between 1.2 and 1.3 million sentence pairs each, development and test datasets are of size 2,000. Detailed token and type statistics can be found in Callison-Burch et al. (2008). The Moses decoder (Koehn et al., 2007) was used for generating lattices and n-best lists. The maximum number of decoding iterations was set to twelve. Since Moses was run with its lexicalised distortion model, there were 14 features. Moses L1-normalises the parameter vector: parameter scaling only marginally affects n-best list construction (via threshold pruning during decoding), while it substantially impacts sampling.

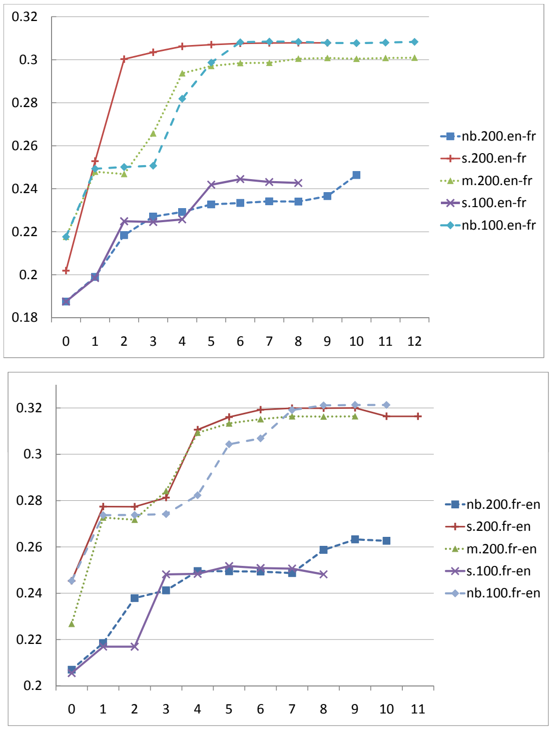


Figure 3: Learning curves (BLEU on the development set) for different tested conditions for English to French (top) and French to English (bottom).

For each of the six configurations, we compared the BLEU score on the test data when optimizing feature weights with MERT using n-best and random samples of size 100 and 200. In all cases we used 20 random restarts for MERT. Results are presented in Table 1. We also ran non systematic experiments on some of the configurations with larger samples and n-best lists, with results changing very little from the respective 200 cases: we do not report them here.

Learning curves (BLEU on the development set) are shown in Figure 3. Learning curves for the other tested language pairs follow a similar pattern.

[4] We assume all phrase pairs cover at least one source word.

5 Analysis of results

All differences of the test scores between optimizing the parameters using nbest-200 lists and from randomly sampled lists of size 200 were found to be statisitically significant at 0.05 level at least. We used Approximate Randomization Test (Riezler and Maxwell, 2005) for the purpose, random sampling being done 1000 times.

S-T	NB-100	RS-100	NB-200	RS-200
En-Fr	32.47	31.36	32.32	**32.76**
Fr-En	32.43	31.77	32.46	**32.91**
En-Es	29.21	28.98	29.65	**30.19**
Es-En	30.97	30.41	31.22	**31.66**
En-De	20.36	19.92	20.55	**20.93**
De-En	27.48	26.98	27.30	**27.62**

Table 1: Test set BLEU Scores for six different “Source-Target” Pairs

Somewhat surprisingly, while random sampling with sample size of 200 yields overall the best results, random sampling with size 100 give systematically worse results than n-best lists of the same size. We conjectured that n-best lists and random samples could have complementary advantages. Indeed, it seems intuitive that a good translation pool should be sufficiently varied, as argued in Section 3.1. However it should also stand high chances to contain the best reachable translation, or translations close to the best. It might thus be that 100-best lists are unable to provide diversity, and random samples of size 100 to guarantee sufficient quality.

In order to test this conjecture we repeated our experiments, but at each iteration we used the union of a 100 random sample and a 100 n-best list. Results for this experiments are in Table 2. The corresponding results with random samples of size 200 are also repeated to ease comparison. Depending on the language pair, improvements over random sampling range from 0.17 (En-Es) to 0.44 (Fr-En) BLEU points. Improvements over 200-best lists range from 0.68 (De-En) to 0.89 (Fr-En) BLEU points. These results indicate quite clearly that N-best lists and random samples contribute complementary information to the translation pool: indeed, in most cases there is very little or no overlap between the two.

Source-Target	Mixed 100 + 100	RS-200
En-Fr	33.17	32.76
Fr-En	33.35	32.91
En-Es	30.37	30.19
Es-En	32.04	31.66
En-De	21.31	20.93
De-En	27.98	27.62

Table 2: Test set BLEU Scores for the same ’‘Source-Target” pairs using a mixed strategy combining a 100 N-best list and a random sample of size 100 after each round of decoding.

Convergence curves show that RS-200, NB-100 and M-200 (i.e. the hybrid combination) systematically converge to higher BLEU scores, on the development set and on their respective translation pools, than RS-100 and NB-200. Notice however that it is misleading to compare scores across different translation pools, especially if these have substantially different sizes. On the one hand adding more candidates increases the chances of adding one with high contribution to the corpus BLEU, and can thus increase the achievable value of the objective function. On the other hand, adding more candidates reduces the freedom MERT has to find parameter values selecting high-BLEU candidates for all sentences. To see this, consider the extreme case when the translation pools are all of size one and are provided by an oracle that gives the highest-BLEU reachable translation for each sentence: the objective surface is uninformatively flat, all values of the parameters are equally good, and the BLEU score on the devset is the highest achievable one. If now we add to each translation pool the second-best BLEU-scoring candidate, BLEU will be maximized in a half-space for each sentence in the development set: MERT will try to select λ in the intersection of all the half-spaces, if this is not empty, but will have to settle for a lower-scoring compromise otherwise. The larger the translation pools, the more difficult it becomes for MERT to “make all sentences happy”. A special case of this is when adding more candidates extends the convex envelopes in such a way that the best candidates fall in the interior of the convex hull. It is difficult to tell which of the two opposing effects (the one that tends to increase the value of the objective function or the one that tends to depress it) is stronger in any given case, but from the convergence curves it would seem that the first prevails in the case of random samples, whereas the second wins in the case of n-best lists. In the case of random samples going from size 100 to 200 systematically leads to higher BLEU score on the devsets, as more high-BLEU candidates are drawn. In the case of n-best lists, conversely, this leads to lower BLEU scores, as lower-BLEU (in average) candidates are added to translation pools providing a sharper representation of the BLEU surface and growing MERT out of the "delusion" that a given high BLEU score is actually achieveable.

In the light of this discussion, it is interesting to observe that the value achieved by the objective function on the development set is only a weak predictor of performance on the test set, e.g. M-200 never converges to values above those of NB-100, but is systematically superior on the test data.

In Macherey et al. (2008) the authors observe a dip in the value of the objective function at the first iteration when training using n-best lists. We did not observe this behaviour in our experiments. A possible explanation for this resides in the larger size of the n-best lists we use (100 or 200, compared to 50 in the cited work) and in the smaller number of dimensions (14 instead of 20-30).

We hinted in Section 3.3 that it would seem reasonable to use samples/nbest-list of size increasing with the length of the source sentence, so as to sample reachable translations with a more uniform density across development sentences. We tested this idea on the French to English condition, making samples size depend linearly on the length of the sentence, and in such a way that the average sample size is either 100 or 200. For average sample size 100 we obtained a BLEU of 31.55 (compared to 31.77 with the constant-size 100 random sample) and for average size 200 31.84 (32.46 in the corresponding constant-size condition). While partial, these results are not particularly encouraging w.r.t. using variable size samples.

Finally, in order to assess the stability of the proposed training procedure across variations in development datasets, we experimented with extracting five distinct devsets of size 2,000 each for the French to English RS-200 condition, keeping the test set fixed: the maximum difference we observed was of 0.33 BLEU points.

6 Conclusions

We introduced a novel variant to the well-known MERT method for performing parameter estimation in Statistical Machine Translation systems based on log-linear models. This method, of straightforward implementation, is based on sampling candidates from the posterior distribution as approximated by an existing translation lattice in order to progressively expand the *translation pool* that shapes the optimization surface. This method compares favorably against existing methods on different accounts. Compared to the standard method by which N-best lists are used to grow the translation pool, it yields empirically better results as shown in our experiments, without significant penalties in terms of computational complexity. These results are in agreement with the intuition that the sampling method introduces more variety in the translation pool, and thus allows to perform more effective parameter updates towards the optimum. A hybrid strategy, consisting in combining N-best lists and random samples, brings about further significant improvements, indicating that both quality and variety are desireable in the translation pool that defines the optimization surface. A possible direction to investigate in the future consists in generalizing this hybrid strategy and combining random samples where the probability distribution induced on the lattice by the current parameters is scaled by a further temperature parameter β:

$$P'(\mathbf{e}, \mathbf{a}|\mathbf{f}) \propto P(\mathbf{e}, \mathbf{a}|\mathbf{f})^{\beta} \tag{7}$$

where for $\beta = 1$ the random samples used in this paper are obtained, for β tending to infinite the distribution becomes peaked around the single best path, thus producing samples similar to N-best lists, and samples from other real values of the temperature can be combined.

Compared to the method using the whole lattice, the proposed approaches have a substantially lower computational complexity under very broad and common assumptions, and yet yield translation quality improvements of comparable magnitude over the baseline N-best list method.

While the method presented in this paper operates on the translation lattices generated by Phrase-Based SMT decoders, the extension to translation forests generated by hierarchical decoders (Chiang, 2007) seems straightforward. In that case, the backward sweep for propagating unnormalized posterior probabilities is replaced by a bottom-up sweep, and the sampling now concerns (binary) trees instead of paths, but the rest of the procedure is substantially unchanged. We conjecture however that the extension to translation forests would be less competitive compared to working with the whole packed forest (as in (Kumar et al., 2009)) than lattice sampling is compared to working with the whole lattice. The reason we believe this is that hierarchical models lead to much more spurious ambiguity than phrase-based models, so that both the N-best method and the sampling method explore a smaller portion of the candidate space compared to the compact representation of all the candidate translations in a beam.

Acknowledgements

We would like to thank Vassilina Nikoulina, Greg Hanneman, Marc Dymetman for useful discussions and the anonymous reviewers for their suggestions and constructive criticism.

References

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. Further meta-evaluation of machine translation. In *Proceedings of the ACL 2008 Workshop on Statistical Machine Translation*, Columbus, Ohio, 2008. http://www.statmt.org/wmt08/pdf/WMT09.pdf.

Daniel Cer, Daniel Jurafsky, and Christopher D. Manning. Regularization and search for minimum error rate training. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 26–34, Columbus, Ohio, 2008. ISBN 978-1-932432-09-1.

David Chiang. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228, 2007.

David Chiang, Yuval Marton, and Philip Resnik. Online large-margin training of syntactic and structural translation features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '08)*, pages 224–233, Honolulu, Hawaii, 2008.

Koby Crammer and Yoram Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research (JMLR)*, 3:951–991, 2003. ISSN 1532-4435. doi: http://dx.doi.org/10.1162/jmlr.2003.3.4-5.951.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research (JMLR)*, 7:551–585, 2006. ISSN 1532-4435.

Kevin Knight. Decoding complexity in word-replacement translation modals. Computational Linguistics, Squibs and Discussion, 25(4),, 1999.

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. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computationl Linguistics (ACL '07)*, pages 177–180, prague, Czech republic, 2007.

Shankar Kumar, Wolfgang Macherey, Chris Dyer, and Franz Och. Efficient minimum error rate training and minimum bayes-risk decoding for translation hypergraphs and lattices. In *Proceedings of the Joint 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 163–171, Suntec, Singapore, August 2009.

Wolfgang Macherey, Franz Josef Och, Ignacio Thayer, and Jakob Uszkoreit. Lattice-based minimum error rate training for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '08)*, pages 725–734, Honolulu, Hawaii, 2008.

Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL '03)*, pages 160–167, Sapporo, Japan, 2003. doi: http://dx.doi.org/10.3115/1075096.1075117.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL '02)*, pages 311–318, Philadelphia, Pennsylvania, 2002. doi: http://dx.doi.org/10.3115/1073083.1073135.

William H. Press. *Numerical recipes : the art of scientific computing*. Cambridge University Press, third edition, September 2007. ISBN 0521880688.

Stefan Riezler and John T. Maxwell. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 57–64, Ann Arbor, Michigan, June 2005.

David A. Smith and Jason Eisner. Minimum risk annealing for training log-linear models. In *Proceedings of the Joint International Conference on Computational Linguistics and Annual meeting of the Association for Computational Linguistics (COLING/ACL '06)*, pages 787–794, Sydney, Australia, 2006.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. Online large-margin training for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 764–773, Prague, Czech Republic, June 2007.

Richard Zens, Sasa Hasan, and Hermann Ney. A systematic comparison of training criteria for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 524–532, Prague, Czech Republic, June 2007.

Statistical Machine Translation with a Factorized Grammar

Libin Shen and **Bing Zhang** and **Spyros Matsoukas** and
Jinxi Xu and **Ralph Weischedel**
Raytheon BBN Technologies
Cambridge, MA 02138, USA
{lshen,bzhang,smatsouk,jxu,weisched}@bbn.com

Abstract

In modern machine translation practice, a statistical phrasal or hierarchical translation system usually relies on a huge set of translation rules extracted from bi-lingual training data. This approach not only results in space and efficiency issues, but also suffers from the sparse data problem. In this paper, we propose to use factorized grammars, an idea widely accepted in the field of linguistic grammar construction, to generalize translation rules, so as to solve these two problems. We designed a method to take advantage of the XTAG English Grammar to facilitate the extraction of factorized rules. We experimented on various setups of low-resource language translation, and showed consistent significant improvement in BLEU over state-of-the-art string-to-dependency baseline systems with 200K words of bi-lingual training data.

1 Introduction

A statistical phrasal (Koehn et al., 2003; Och and Ney, 2004) or hierarchical (Chiang, 2005; Marcu et al., 2006) machine translation system usually relies on a very large set of translation rules extracted from bi-lingual training data with heuristic methods on word alignment results. According to our own experience, we obtain about 200GB of rules from training data of about 50M words on each side. This immediately becomes an engineering challenge on space and search efficiency.

A common practice to circumvent this problem is to filter the rules based on development sets in the step of rule extraction or before the decoding phrase, instead of building a real distributed system. However, this strategy only works for research systems, for which the segments for translation are always fixed.

However, do we really need such a large rule set to represent information from the training data of much smaller size? Linguists in the grammar construction field already showed us a perfect solution to a similar problem. The answer is to use a factorized grammar. Linguists decompose lexicalized linguistic structures into two parts, (unlexicalized) templates and lexical items. Templates are further organized into families. Each family is associated with a set of lexical items which can be used to lexicalize all the templates in this family. For example, the XTAG English Grammar (XTAG-Group, 2001), a hand-crafted grammar based on the Tree Adjoining Grammar (TAG) (Joshi and Schabes, 1997) formalism, is a grammar of this kind, which employs factorization with LTAG e-tree templates and lexical items.

Factorized grammars not only relieve the burden on space and search, but also alleviate the sparse data problem, especially for low-resource language translation with few training data. With a factored model, we do not need to observe exact "template – lexical item" occurrences in training. New rules can be generated from template families and lexical items either offline or on the fly, explicitly or implicitly. In fact, the factorization approach has been successfully applied on the morphological level in previous study on MT (Koehn and Hoang, 2007). In this work, we will go further to investigate factorization of rule structures by exploiting the rich XTAG English Grammar.

We evaluate the effect of using factorized translation grammars on various setups of low-resource language translation, since low-resource MT suffers greatly on poor generalization capability of trans-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 616–625,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

lation rules. With the help of high-level linguistic knowledge for generalization, factorized grammars provide consistent significant improvement in BLEU (Papineni et al., 2001) over string-to-dependency baseline systems with 200K words of bi-lingual training data.

This work also closes the gap between compact hand-crafted translation rules and large-scale unorganized automatic rules. This may lead to a more effective and efficient statistical translation model that could better leverage generic linguistic knowledge in MT.

In the rest of this paper, we will first provide a short description of our baseline system in Section 2. Then, we will introduce factorized translation grammars in Section 3. We will illustrate the use of the XTAG English Grammar to facilitate the extraction of factorized rules in Section 4. Implementation details are provided in Section 5. Experimental results are reported in Section 6.

2 A Baseline String-to-Tree Model

As the baseline of our new algorithm, we use a string-to-dependency system as described in (Shen et al., 2008). There are several reasons why we take this model as our baseline. First, it uses syntactic tree structures on the target side, which makes it easy to exploit linguistic information. Second, dependency structures are relatively easier to implement, as compared to phrase structure grammars. Third, a string-to-dependency system provides state-of-the-art performance on translation accuracy, so that improvement over such a system will be more convincing.

Here, we provide a brief description of the baseline string-to-dependency system, for the sake of completeness. Readers can refer to (Shen et al., 2008; Shen et al., 2009) for related information.

In the baseline string-to-dependency model, each translation rule is composed of two parts, source and target. The source sides is a string rewriting rule, and the target side is a tree rewriting rule. Both sides can contain non-terminals, and source and target non-terminals are one-to-one aligned. Thus, in the decoding phase, non-terminal replacement for both sides are synchronized.

Decoding is solved with a generic chart parsing algorithm. The source side of a translation rule is used to detect when this rule can be applied. The target side of the rule provides a hypothesis tree structure for the matched span. Mono-lingual parsing can be viewed as a special case of this generic algorithm, for which the source string is a projection of the target tree structure.

Figure 1 shows three examples of string-to-dependency translation rules. For the sake of convenience, we use English for both source and target. Upper-cased words represent source, while lower-cased words represent target. X is used for non-terminals for both sides, and non-terminal alignment is represented with subscripts.

In Figure 1, the top boxes mean the source side, and the bottom boxes mean the target side. As for the third rule, *FUN_Q* stands for a function word in the source language that represents a question.

3 Translation with a Factorized Grammar

We continue with the example rules in Figure 1. Suppose, we have "... *HATE* ... *FUN_Q*" in a given test segment. There is no rule having both *HATE* and *FUN_Q* on its source side. Therefore, we have to translate these two source words separately. For example, we may use the second rule in Figure 1. Thus, *HATE* will be translated into *hates*, which is wrong.

Intuitively, we would like to have translation rule that tell us how to translate *X1 HATE X2 FUN_Q* as in Figure 2. It is not available directly from the training data. However, if we obtain the three rules in Figure 1, we are able to predict this missing rule. Furthermore, if we know *like* and *hate* are in the same syntactic/semantic class in the source or target language, we will be very confident on the validity of this hypothesis rule.

Now, we propose a factorized grammar to solve this generalization problem. In addition, translation rules represented with the new formalism will be more compact.

3.1 Factorized Rules

We decompose a translation rule into two parts, a pair of **lexical items** and an unlexicalized **template**. It is similar to the solution in the XTAG English Grammar (XTAG-Group, 2001), while here we

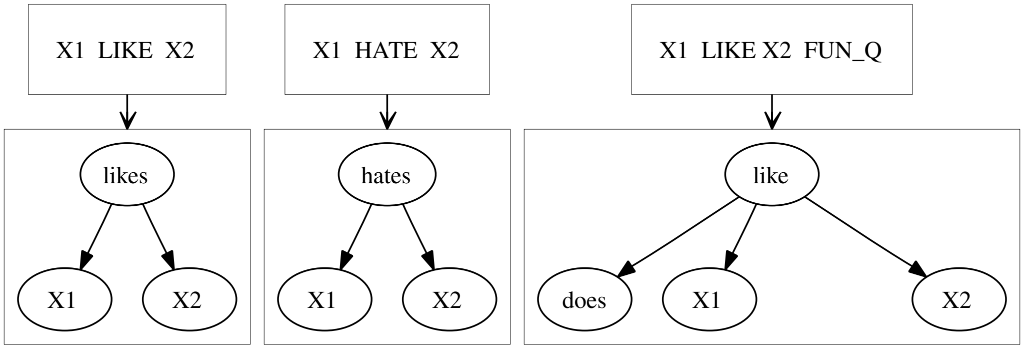


Figure 1: Three examples of string-to-dependency translation rules.

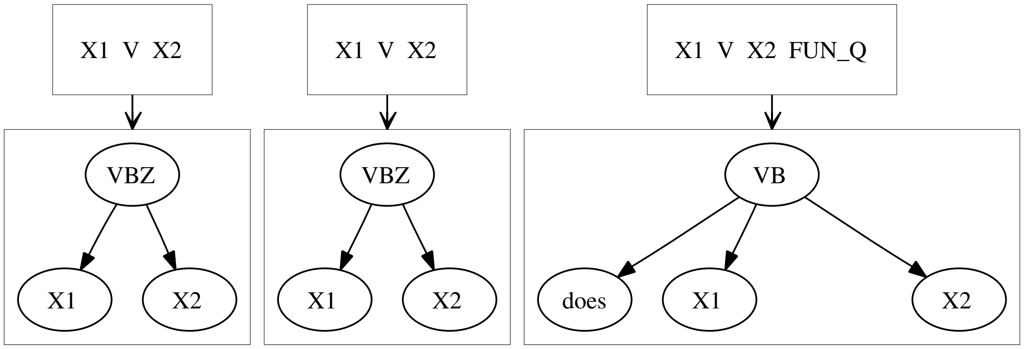


Figure 3: Templates for rules in Figure 1.

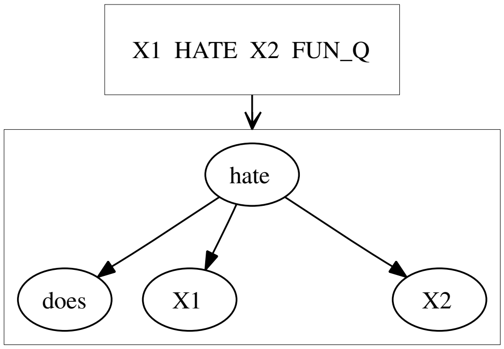


Figure 2: An example of a missing rule.

work on two languages at the same time.

For each rule, we first detect a pair of aligned head words. Then, we extract the stems of this word pair as lexical items, and replace them with their POS tags in the rule. Thus, the original rule becomes an unlexicalized rule template.

As for the three example rules in Figure 1, we will extract lexical items (*LIKE, like*), (*HATE, hate*) and (*LIKE, like*) respectively. We obtain the same lexical items from the first and the third rules.

The resultant templates are shown in Figure 3. Here, *V* represents a verb on the source side, VB stands for a verb in the base form, and *VBZ* means a verb in the third person singular present form as in the Penn Treebank representation (Marcus et al., 1994).

In the XTAG English Grammar, tree templates for transitive verbs are grouped into a family. All transitive verbs are associated with this family. Here, we assume that the rule templates representing structural variations of the same word class can also be organized into a template **family**. For example, as shown in Figure 4, templates and lexical items are associated with families. It should be noted that a template or a lexical item can be associated with more than one family.

Another level of indirection like this provides more generalization capability. As for the missing

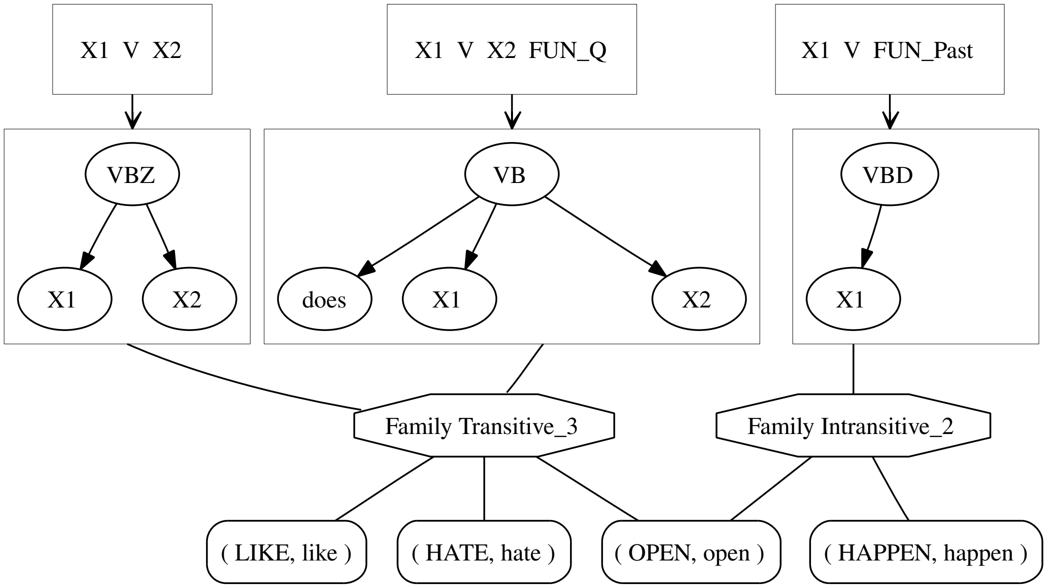


Figure 4: Templates and lexical items are associated with families.

rule in Figure 2, we can now generate it by replacing the POS tags in the second template of Figure 4 with lexical items (*HATE, hate*) with their correct inflections. Both the template and the lexical items here are associated with the family Transitive_3..

3.2 Statistical Models

Another level of indirection also leads to a desirable back-off model. We decompose a rule R into to two parts, its template P_R and its lexical items L_R. Assuming they are independent, then we can compute $Pr(R)$ as

$$Pr(R) = Pr(P_R)Pr(L_R), \text{ or}$$
$$Pr(R) = \textstyle\sum_F Pr(P_R|F)Pr(L_R|F)Pr(F), \quad (1)$$

if they are conditionally independent for each family F. In this way, we can have a good estimate for rules that do not appear in the training data. The second generative model will also be useful for unsupervised learning of families and related probabilities.

In this paper, we approximate families by using target (English) side linguistic knowledge as what we will explain in Section 4, so this changes the definition of the task. In short, we will be given a list of families. We will also be given an association table $B(L, F)$ for lexical items L and families F, such that $B(L, F) = true$ if and only L is associated with F, but we do not know the distributions.

Let S be the source side of a rule or a rule template, T the target side of a rule of a rule template. We define Pr_b, the back-off conditional model of templates, as follows.

$$Pr_b(P_S|P_T, L) = \frac{\sum_{F:B(L,F)} \#(P_S, P_T, F)}{\sum_{F:B(L,F)} \#(P_T, F)}, \quad (2)$$

where # stands for the count of events.

Let P and L be the template and lexical items of R respectively. Let Pr_t be the MLE model obtained from the training data. The smoothed probability is then defined as follows.

$$\begin{aligned} Pr(R_S|R_T) &= (1-\alpha)Pr_t(R_S|R_T) \\ &\quad +\alpha Pr_b(P_S|P_T, L), \quad (3) \end{aligned}$$

where α is a parameter. We fix it to 0.1 in later experiments. Conditional probability $Pr(R_T|R_S)$ is defined in a similar way.

3.3 Discussion

The factorized models discussed in the previous section can greatly alleviate the sparse data problem, especially for low-resource translation tasks. However, when the training data is small, it is not easy to learn families. Therefore, to use unsupervised learning with a model like (1) somehow reduces a hard translation problem to another one of the same difficulty, when the training data is small.

However, in many cases, we do have extra information that we can take advantage of. For example, if the target language has rich resources, although the source language is a low-density one, we can exploit the linguistic knowledge on the target side, and carry it over to bi-lingual structures of the translation model. The setup of X-to-English translation tasks is just like this. This will be the topic of the next section. We leave unsupervised learning of factorized translation grammars for future research.

4 Using A Mono-Lingual Grammar

In this section, we will focus on X-to-English translation, and explain how to use English resources to build a factorized translation grammar. Although we use English as an example, this approach can be applied to any language pairs that have certain linguistic resources on one side.

As shown in Figure 4, intuitively, the families are intersection of the word families of the two languages involved, which means that they are refinement of the English word families. For example, a sub-set of the English transitive families may be translated in the same way, so they share the same set of templates. This is why we named the two families Transitive_3 and Intransitive_2 in Figure 4.

Therefore, we approximate bi-lingual families with English families first. In future, we can use them as the initial values for unsupervised learning.

In order to learn English families, we need to take away the source side information in Figure 4, and we end up with a template–family–word graph as shown in Figure 5. We can learn this model on large mono-lingual data if necessary.

What is very interesting is that there already exists a hand-crafted solution for this model. This is the XTAG English Grammar (XTAG-Group, 2001).

The XTAG English Grammar is a large-scale English grammar based on the TAG formalism extended with lexicalization and unification-based feature structures. It consists of morphological, syntactic, and tree databases. The syntactic database contains the information that we have represented in Figure 5 and many other useful linguistic annotations, e.g. features.

The XTAG English grammar contains 1,004 templates, organized in 53 families, and 221 individual templates. About 30,000 lexical items are associated with these families and individual templates [1]. In addition, it also has the richest English morphological lexicon with 317,000 inflected items derived from 90,000 stems. We use this resource to predict POS tags and inflections of lexical items.

In our applications, we select all the verb families plus one each for nouns, adjectives and adverbs. We use the families of the English word as the families of bi-lingual lexical items. Therefore, we have a list of about 20 families and an association table as described in Section 3.2. Of course, one can use other linguistic resources if similar family information is provided, e.g. VerbNet (Kipper et al., 2006) or WordNet (Fellbaum, 1998).

5 Implementation

Nowadays, machine translation systems become more and more complicated. It takes time to write a decoder from scratch and hook it with various modules, so it is not the best solution for research purpose. A common practice is to reduce a new translation model to an old one, so that we can use an existing system, and see the effect of the new model quickly. For example, the tree-based model proposed in (Carreras and Collins, 2009) used a phrasal decoder for sub-clause translation, and recently, DeNeefe and Knight (2009) reduced a TAG-based translation model to a CFG-based model by applying all possible adjunction operations offline and stored the results as rules, which were then used by an existing syntax-based decoder.

Here, we use a similar method. Instead of building a new decoder that uses factorized grammars, we reduce factorized rules to baseline string-to-dependency rules by performing combination of templates and lexical items in an offline mode. This is similar to the rule generation method in (DeNeefe and Knight, 2009). The procedure is as follows.

In the rule extraction phase, we first extract all the string-to-dependency rules with the baseline system.

[1] More information about XTAG is available online at http://www.cis.upenn.edu/˜xtag .

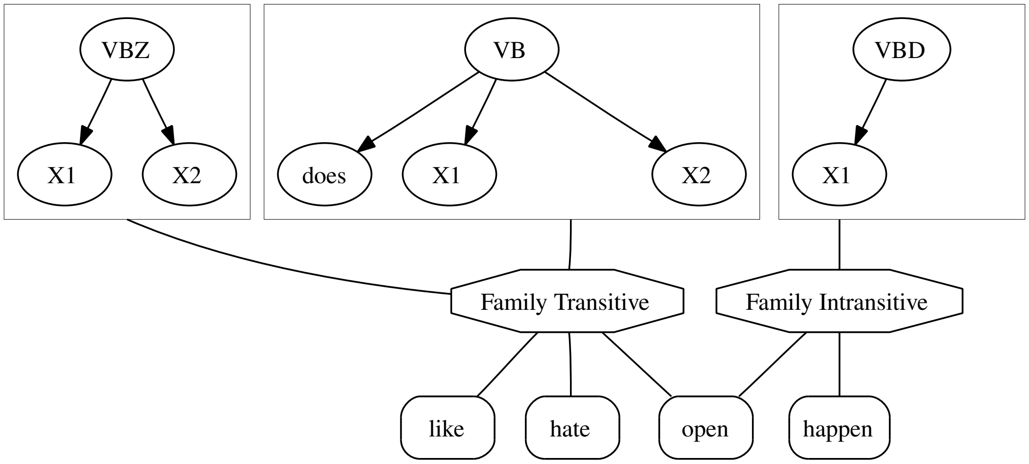


Figure 5: Templates, families, and words in the XTAG English Grammar.

For each extracted rule, we try to split it into various "template–lexical item" pairs by choosing different aligned words for delexicalization, which turns rules in Figure 1 into lexical items and templates in Figure 3. Events of templates and lexical items are counted according to the family of the target English word. If an English word is associated with more than one family, the count is distributed uniformly among these families. In this way, we collect sufficient statistics for the back-off model in (2).

For each family, we keep the top 200 most frequent templates. Then, we apply them to all the lexical items in this families, and save the generated rules. We merge the new rules with the original one. The conditional probabilities for the rules in the combined set is smoothed according to (2) and (3).

Obviously, using only the 200 most frequent templates for each family is just a rough approximation. An exact implementation of a new decoder for factorized grammars can make better use of all the templates. However, the experiments will show that even an approximation like this can already provide significant improvement on small training data sets, i.e. with no more than 2M words.

Since we implement template application in an offline mode, we can use exactly the same decoding and optimization algorithms as the baseline. The decoder is a generic chart parsing algorithm that generates target dependency trees from source string input. The optimizer is an L-BFGS algorithm that maximizes expected BLEU scores on n-best hypotheses (Devlin, 2009).

6 Experiments on Low-Resource Setups

We tested the performance of using factorized grammars on low-resource MT setups. As what we noted above, the sparse data problem is a major issue when there is not enough training data. This is one of the cases that a factorized grammar would help.

We did not tested on real low-resource languages. Instead, we mimic the low-resource setup with two of the most frequently used language pairs, Arabic-to-English and Chinese-to-English, on newswire and web genres. Experiments on these setups will be reported in Section 6.1. Working on a language which actually has more resources allows us to study the effect of training data size. This will be reported in Section 6.2. In Section 6.3, we will show examples of templates learned from the Arabic-to-English training data.

6.1 Languages and Genres

The Arabic-to-English training data contains about 200K (target) words randomly selected from an LDC corpus, LDC2006G05 A2E set, plus an Arabic-English dictionary with about 89K items. We build our development sets from GALE P4 sets. There are one tune set and two test sets for the MT systems [2]. TEST-1 has about 5000 segments and TEST-2 has about 3000 segments.

[2] One of the two test sets will later be used to tune an MT combination system.

MODEL	TUNE			TEST-1			TEST-2		
	BLEU	**%BL**	**MET**	**BLEU**	**%BL**	**MET**	**BLEU**	**%BL**	**MET**
Arabic-to-English newswire									
baseline	21.07	12.41	43.77	19.96	11.42	42.79	21.09	11.03	43.74
factorized	21.70	13.17	44.85	20.52	11.70	43.83	21.36	11.77	44.72
Arabic-to-English web									
baseline	10.26	5.02	32.78	9.40	4.87	31.26	14.11	7.34	35.93
factorized	10.67	5.34	33.83	9.74	5.20	32.52	14.66	7.69	37.11
Chinese-to-English newswire									
baseline	13.17	8.04	44.70	19.62	9.32	48.60	14.53	6.82	45.34
factorized	13.91	8.09	45.03	20.48	9.70	48.61	15.16	7.37	45.31
Chinese-to-English web									
baseline	11.52	5.96	42.18	11.44	6.07	41.90	9.83	4.66	39.71
factorized	11.98	6.31	42.84	11.72	5.88	42.55	10.25	5.34	40.34

Table 1: Experimental results on Arabic-to-English / Chinese-to-English newswire and web data. **%BL** stands for BLEU scores for documents whose BLEU scores are in the bottom 75% to 90% range of all documents. **MET** stands for METEOR scores.

The Chinese-to-English training data contains about 200K (target) words randomly selected from LDC2006G05 C2E set, plus a Chinese-English dictionary (LDC2002L27) with about 68K items. The development data setup is similar to that of Arabic-to-English experiments.

Chinese-to-English translation is from a morphology poor language to a morphology rich language, while Arabic-to-English translation is in the opposite direction. It will be interesting to see if factorized grammars help on both cases. Furthermore, we also test on two genres, newswire and web, for both languages.

Table 1 lists the experimental results of all the four conditions. The tuning metric is expected BLEU. We are also interested in the BLEU scores for documents whose BLEU scores are in the bottom 75% to 90% range of all documents. We mark it as %BL in the table. This metric represents how a system performances on difficult documents. It is important to certain percentile evaluations. We also measure METEOR (Banerjee and Lavie, 2005) scores for all systems.

The system using factorized grammars shows BLEU improvement in all conditions. We measure the significance of BLEU improvement with paired bootstrap resampling as described by (Koehn, 2004). All the BLEU improvements are over 95% confidence level. The new system also improves %BL and METEOR in most of the cases.

6.2 Training Data Size

The experiments to be presented in this section are designed to measure the effect of training data size. We select Arabic web for this set of experiments. Since the original Arabic-to-English training data LDC2006G05 is a small one, we switch to LDC2006E25, which has about 3.5M target words in total. We randomly select 125K, 250K, 500K, 1M and 2M sub-sets from the whole data set. A larger one always includes a smaller one. We still tune on expected BLEU, and test on BLEU, %BL and METEOR.

The average BLEU improvement on test sets is about 0.6 on the 125K set, but it gradually diminishes. For better observation, we draw the curves of BLEU improvement along with significance test results for each training set. As shown in Figure 6 and 7, more improvement is observed with fewer training data. This fits well with fact that the baseline MT model suffers more on the sparse data problem with smaller training data. The reason why the improvement diminishes on the full data set could be that the rough approximation with 200 most frequent templates cannot fully take advantage of this paradigm, which will be discussed in the next section.

MODEL	SIZE	TUNE			TEST-1			TEST-2		
		BLEU	%BL	MET	BLEU	%BL	MET	BLEU	%BL	MET
Arabic-to-English web										
baseline	125K	8.54	2.96	28.87	7.41	2.82	26.95	11.29	5.06	31.37
factorized		8.99	3.44	30.40	7.92	3.57	28.63	12.04	6.06	32.87
baseline	250K	10.18	4.70	32.21	8.94	4.35	30.31	13.71	6.93	35.14
factorized		10.57	4.96	33.22	9.34	4.78	31.51	14.02	7.28	36.25
baseline	500K	12.18	5.84	35.59	10.82	5.77	33.62	16.48	8.30	38.73
factorized		12.40	6.01	36.15	11.14	5.96	34.38	16.76	8.53	39.27
baseline	1M	13.95	7.17	38.49	12.48	7.12	36.56	18.86	10.00	42.18
factorized		14.14	7.41	38.99	12.66	7.34	37.14	19.11	10.29	42.56
baseline	2M	15.74	8.38	41.15	14.18	8.17	39.26	20.96	11.95	45.18
factorized		15.92	8.81	41.51	14.34	8.25	39.68	21.42	12.05	45.51
baseline	3.5M	16.95	9.76	43.03	15.47	9.08	41.28	22.83	13.24	47.05
factorized		17.07	9.99	43.18	15.49	8.77	41.41	22.72	13.10	47.23

Table 2: Experimental results on Arabic web. **%BL** stands for BLEU scores for documents whose BLEU scores are in the bottom 75% to 90% range of all documents. **MET** stands for METEOR scores.

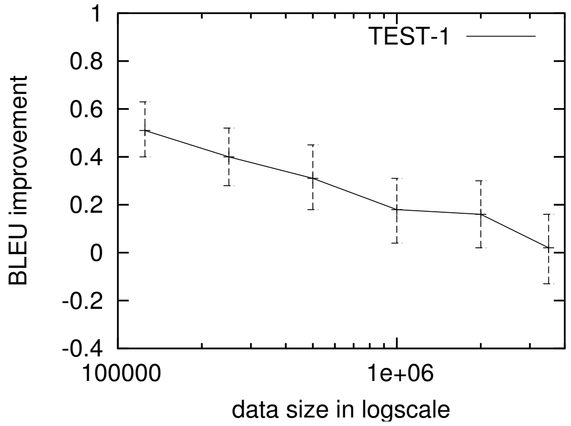


Figure 6: BLEU Improvement with 95% confidence range by using factorized grammars on TEST-1.

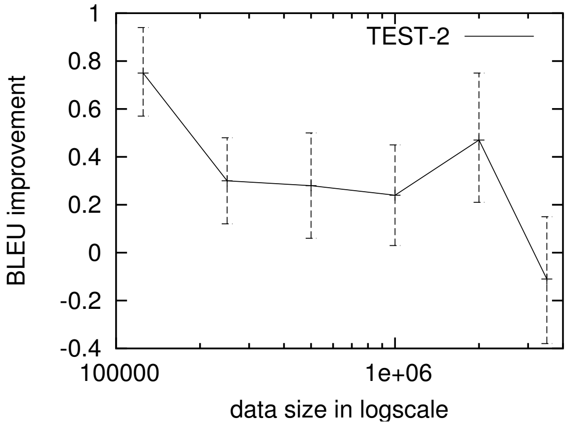


Figure 7: BLEU Improvement with 95% confidence range by using factorized grammars on TEST-2.

6.3 Example Templates

Figure 8 lists seven Arabic-to-English templates randomly selected from the transitive verb family. TMPL_151 is an interesting one. It helps to alleviate the pronoun dropping problem in Arabic. However, we notice that most of the templates in the 200 lists are rather simple. More sophisticated solutions are needed to go deep into the list to find out better templates in future.

It will be interesting to find an automatic or semi-automatic way to discover source counterparts of target treelets in the XTAG English Grammar. Generic rules like this will be very close to handcraft translate rules that people have accumulated for rule-based MT systems.

7 Conclusions and Future Work

In this paper, we proposed a novel statistical machine translation model using a factorized structure-based translation grammar. This model not only alleviates the sparse data problem but only relieves the burden on space and search, both of which are imminent issues for the popular phrasal and/or hierarchical MT systems.

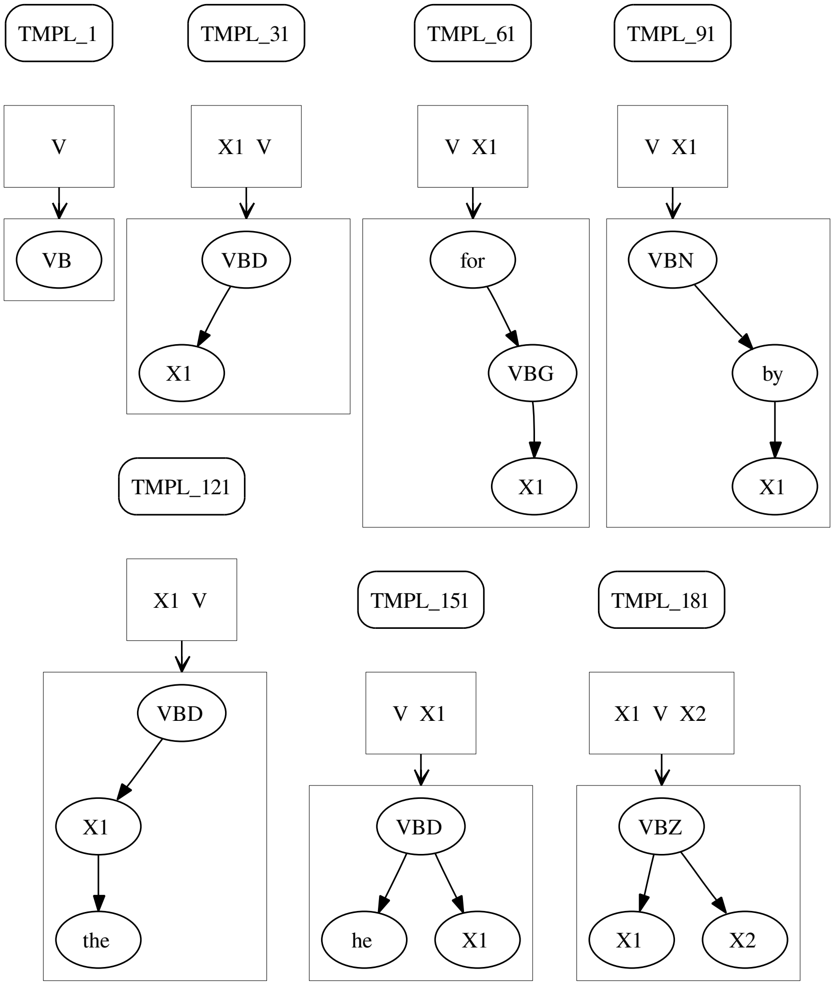


Figure 8: Randomly selected Arabic-to-English templates from the transitive verb family.

We took low-resource language translation, especially X-to-English translation tasks, for case study. We designed a method to exploit family information in the XTAG English Grammar to facilitate the extraction of factorized rules. We tested the new model on low-resource translation, and the use of factorized models showed significant improvement in BLEU on systems with 200K words of bi-lingual training data of various language pairs and genres.

The factorized translation grammar proposed here shows an interesting way of using richer syntactic resources, with high potential for future research.

In future, we will explore various learning methods for better estimation of families, templates and lexical items. The target linguistic knowledge that we used in this paper will provide a nice starting point for unsupervised learning algorithms.

We will also try to further exploit the factorized representation with discriminative learning. Features defined on templates and families will have good generalization capability.

Acknowledgments

This work was supported by DARPA/IPTO Contract HR0011-06-C-0022 under the GALE program[3]. We thank Aravind Joshi, Scott Miller, Richard Schwartz and anonymous reviewers for valuable comments.

[3] Distribution Statement ”A” (Approved for Public Release, Distribution Unlimited). The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

References

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the 43th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 101–104, Ann Arbor, MI.

Xavier Carreras and Michael Collins. 2009. Non-projective parsing for statistical machine translation. In *Proceedings of the 2009 Conference of Empirical Methods in Natural Language Processing*, pages 200–209, Singapore.

David 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)*, pages 263–270, Ann Arbor, MI.

Steve DeNeefe and Kevin Knight. 2009. Synchronous tree adjoining machine translation. In *Proceedings of the 2009 Conference of Empirical Methods in Natural Language Processing*, pages 727–736, Singapore.

Jacob Devlin. 2009. Lexical features for statistical machine translation. Master's thesis, Univ. of Maryland.

Christiane Fellbaum, editor. 1998. *WordNet: an electronic lexical database*. The MIT Press.

Aravind K. Joshi and Yves Schabes. 1997. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 69–124. Springer-Verlag.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2006. Extensive classifications of english verbs. In *Proceedings of the 12th EURALEX International Congress*.

P. Koehn and H. Hoang. 2007. Factored translation models. In *Proceedings of the 2007 Conference of Empirical Methods in Natural Language Processing*.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 48–54, Edmonton, Canada.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference of Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. SPMT: Statistical machine translation with syntactified target language phrases. In *Proceedings of the 2006 Conference of Empirical Methods in Natural Language Processing*, pages 44–52, Sydney, Australia.

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.

Franz J. Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4).

Kishore Papineni, Salim Roukos, and Todd Ward. 2001. Bleu: a method for automatic evaluation of machine translation. IBM Research Report, RC22176.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Libin Shen, Jinxi Xu, Bing Zhang, Spyros Matsoukas, and Ralph Weischedel. 2009. Effective Use of Linguistic and Contextual Information for Statistical Machine Translation. In *Proceedings of the 2009 Conference of Empirical Methods in Natural Language Processing*, pages 72–80, Singapore.

XTAG-Group. 2001. A lexicalized tree adjoining grammar for english. Technical Report 01-03, IRCS, Univ. of Pennsylvania.

Discriminative Sample Selection for Statistical Machine Translation[*]

Sankaranarayanan Ananthakrishnan, Rohit Prasad, David Stallard, and Prem Natarajan
Raytheon BBN Technologies
10 Moulton Street
Cambridge, MA, U.S.A.
{sanantha,rprasad,stallard,prem}@bbn.com

Abstract

Production of parallel training corpora for the development of statistical machine translation (SMT) systems for resource-poor languages usually requires extensive manual effort. Active sample selection aims to reduce the labor, time, and expense incurred in producing such resources, attaining a given performance benchmark with the smallest possible training corpus by choosing informative, non-redundant source sentences from an available candidate pool for manual translation. We present a novel, discriminative sample selection strategy that preferentially selects batches of candidate sentences with constructs that lead to erroneous translations on a held-out development set. The proposed strategy supports a built-in diversity mechanism that reduces redundancy in the selected batches. Simulation experiments on English-to-Pashto and Spanish-to-English translation tasks demonstrate the superiority of the proposed approach to a number of competing techniques, such as random selection, dissimilarity-based selection, as well as a recently proposed semi-supervised active learning strategy.

1 Introduction

Resource-poor language pairs present a significant challenge to the development of statistical machine translation (SMT) systems due to the latter's dependence on large parallel texts for training. Bilingual human experts capable of producing the requisite data resources are often in short supply, and the task of preparing high-quality parallel corpora is laborious and expensive. In light of these constraints, an attractive strategy is to construct the smallest possible parallel training corpus with which a desired performance benchmark may be achieved.

Such a corpus may be constructed by selecting the most informative instances from a large collection of source sentences for translation by a human expert, a technique often referred to as *active learning*. A SMT system trained with sentence pairs thus generated is expected to perform significantly better than if the source sentences were chosen using, say, a naïve random sampling strategy.

Previously, Eck et al. (2005) described a selection strategy that attempts to maximize coverage by choosing sentences with the highest proportion of previously unseen n-grams. Depending on the composition of the candidate pool with respect to the domain, this strategy may select irrelevant outliers. They also described a technique based on TF-IDF to de-emphasize sentences similar to those that have already been selected, thereby encouraging diversity. However, this strategy is bootstrapped by random initial choices that do not necessarily favor sentences that are difficult to translate. Finally, they worked exclusively with the source language and did not use any SMT-derived features to guide selection.

Haffari et al. (2009) proposed a number of features, such as similarity to the seed corpus, translation probability, n-gram and phrase coverage, etc., that drive data selection. They also proposed a model in which these features combine linearly to predict a rank for each candidate sentence. The

[*] Distribution Statement "A" (Approved for Public Release, Distribution Unlimited)

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 626–635,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

top-ranked sentences are chosen for manual translation. However, this approach requires that the pool have the same distributional characteristics as the development sets used to train the ranking model. Additionally, batches are chosen atomically. Since similar or identical sentences in the pool will typically meet the selection criteria simultaneously, this can have the undesired effect of choosing redundant batches with low diversity.

The semi-supervised active learning strategy proposed by Ananthakrishnan et al. (2010) uses multilayer perceptrons (MLPs) to rank candidate sentences based on various features, including domain representativeness, translation difficulty, and batch diversity. A greedy, incremental batch construction technique encourages diversity. While this strategy was shown to be superior to random as well as n-gram based dissimilarity selection, its coarse granularity (reducing a candidate sentence to a low-dimensional feature vector for ranking) makes it unsuitable for many situations. In particular, it is seen to have little or no benefit over random selection when there is no logical separation of the candidate pool into "in-domain" and "out-of-domain" subsets.

This paper introduces a novel, active sample selection technique that identifies translation errors on a held-out development set, and preferentially selects candidate sentences with constructs that are incorrectly translated in the former. A discriminative pairwise comparator function, trained on the ranked development set, is used to order candidate sentences and pick sentences that provide maximum potential reduction in translation error. The feature functions that power the comparator are updated after each selection to encourage batch diversity. In the following sections, we provide details of the proposed sample selection approach, and describe simulation experiments that demonstrate its superiority over a number of competing strategies.

2 Error-Driven Active Learning

Traditionally, unsupervised selection strategies have dominated the active learning literature for natural language processing (Hwa, 2004; Tang et al., 2002; Shen et al., 2004). Sample selection for SMT has followed a similar trend. The work of Eck et al. (2005) and most of the techniques proposed by Haffari et al. (2009) fall in this category. Notable exceptions include the linear ranking model of Haffari et al. (2009) and the semi-supervised selection technique of Ananthakrishnan et al. (2010), both of which use one or more held-out development sets to train and tune the sample selector. However, while the former uses the posterior translation probability and the latter, a sentence-level confidence score as part of the overall selection strategy, current active learning techniques for SMT do not explicitly target the sources of error.

Error-driven active learning attempts to choose candidate instances that potentially maximize error reduction on a reference set (Cohn et al., 1996; Meng and Lee, 2008). In the context of SMT, this involves decoding a held-out development set with an existing baseline (seed) SMT system. The selection algorithm is then trained to choose, from the candidate pool, sentences containing constructs that give rise to translation errors on this set. Assuming perfect reference translations and word alignment in subsequent SMT training, these sentences provide maximum potential reduction in translation error with respect to the seed SMT system. It is a *supervised* approach to sample selection. We assume the following are available.

- A seed parallel corpus $\mathbf{S}$ for training the initial SMT system.
- A candidate pool of monolingual source sentences $\mathbf{P}$ from which samples must be selected.
- A held-out development set $\mathbf{D}$ for training the selection algorithm and for tuning the SMT.
- A test set $\mathbf{T}$ for evaluating SMT performance.

We further make the following reasonable assumptions: (a) the development set $\mathbf{D}$ and the test set $\mathbf{T}$ are drawn from the same distribution and (b) the candidate pool $\mathbf{P}$ consists of both in- and out-of-domain source sentences, as well as an allowable level of redundancy (similar or identical sentences).

Using translation errors on the development set to drive sample selection has the following advantages over previously proposed active learning strategies for SMT.

- The seed training corpus $\mathbf{S}$ need not be derived from the same distribution as $\mathbf{D}$ and $\mathbf{T}$. The seed SMT system can be trained with any available parallel corpus for the specified language pair. This is very useful if, as is often the case, little or no in-domain training data is available to bootstrap the SMT system. This removes a critical restriction present in the semi-supervised approach of Ananthakrishnan et al. (2010).

- Sentences chosen are guaranteed to be relevant to the domain, because selection is based on n-grams derived from the development set. This alleviates potential problems with approaches suggested by Eck et al. (2005) and several techniques used by Haffari et al. (2009), where irrelevant outliers may be chosen simply because they contain previously unseen n-grams, or are deemed difficult to translate.
- The proposed technique seeks to minimize held-out translation error rather than maximize training-set coverage. This is the more intuitive, direct approach to sample selection for SMT.
- Diversity can be encouraged by preventing n-grams that appear in previously selected sentences from playing a role in choosing subsequent sentences. This provides an efficient alternative to the cumbersome "batch diversity" feature proposed by Ananthakrishnan et al. (2010).

The proposed implementation of error-driven active learning for SMT, discriminative sample selection, is described in the following section.

3 Discriminative Sample Selection

The goal of active sample selection is to induce an ordering of the candidate instances that satisfies an objective criterion. Eck et al. (2005) ordered candidate sentences based on the frequency of unseen n-grams. Haffari et al. (2009) induced a ranking based on unseen n-grams, translation difficulty, etc., as well as one that attempted to incrementally maximize BLEU using two held-out development sets. Ananthakrishnan et al. (2010) attempted to order the candidate pool to incrementally maximize source n-gram coverage on a held-out development set, subject to difficulty and diversity constraints.

In the case of error-driven active learning, we attempt to learn an ordering model based on errors observed on the held-out development set $\mathbf{D}$. We achieve this in an innovative fashion by casting the ranking problem as a pairwise sentence comparison problem. This approach, inspired by Ailon and Mohri (2008), involves the construction of a binary classifier functioning as a relational operator that can be used to order the candidate sentences. The pairwise comparator is trained on an ordering of $\mathbf{D}$ that ranks constituent sentences in decreasing order of the number of translation errors. The comparator is then used to rank the candidate pool in decreasing order of potential translation error reduction.

3.1 Maximum-Entropy Pairwise Comparator

Given a pair of source sentences (u, v), we define, adopting the notation of Ailon and Mohri (2008), the pairwise comparator $h(u, v)$ as follows:

$$h(u,v) = \begin{cases} 1, & u < v \\ 0, & u >= v \end{cases} \qquad (1)$$

In Equation 1, the binary comparator $h(u, v)$ plays the role of the "less than" ("$<$") relational operator, returning 1 if u is preferred to v in an ordered list, and 0 otherwise. As detailed in Ailon and Mohri (2008), the comparator must satisfy the constraint that $h(u, v)$ and $h(v, u)$ be complementary, i.e. $h(u, v) + h(v, u) = 1$ to avoid ambiguity. However, it need not satisfy the triangle inequality.

We implement $h(u, v)$ as a combination of discriminative maximum entropy classifiers triggered by feature functions drawn from n-grams of u and v. We define $p(u, v)$ as the conditional posterior probability of the Bernoulli event $u < v$ given (u, v) as shown in Equation 2.

$$p(u,v) = Pr(u < v \mid u, v) \qquad (2)$$

In our implementation, $p(u, v)$ is the output of a binary maximum-entropy classifier trained on the development set. However, this implementation poses two problems.

First, if we use constituent n-grams of u and v as feature functions to trigger the classifier, there is no way to distinguish between (u, v) and (v, u) as they will trigger the same feature functions. This will result in identical values for $p(u, v)$ and $p(v, u)$, a contradiction. We resolve this issue by introducing a set of "complementary" feature functions, which are formed by simply appending a recognizable identifier to the existing n-gram feature func-

u: `how are you`
v: `i am going`

$\mathbf{f}(u) = \{$`how:1, are:1, you:1, how*are:2, are*you:2, how*are*you:3`$\}$
$\mathbf{f}(v) = \{$`i:1, am:1, going:1, i*am:2, am*going:2, i*am*going:3`$\}$

$\mathbf{f}'(u) = \{$`!how:1, !are:1, !you:1, !how*are:2, !are*you:2, !how*are*you:3`$\}$
$\mathbf{f}'(v) = \{$`!i:1, !am:1, !going:1, !i*am:2, !am*going:2, !i*am*going:3`$\}$

Table 1: Standard and complementary trigram feature functions for a source pair (u, v).

tions. Then, to evaluate $p(u, v)$, for instance, we invoke the classifier with standard feature functions for u and complementary feature functions for v. Similarly, $p(v, u)$ is evaluated by triggering complementary feature functions for u and standard feature functions for v. Table 1 illustrates this with a simple example.

Note that each feature function is associated with a real value, whose magnitude is an indicator of its importance. In our implementation, an n-gram feature function (standard or complementary) receives a value equal to its length. This is based on our intuition that longer n-grams play a more important role in dictating SMT performance.

Second, the introduction of complementary triggers implies that evaluation of $p(u, v)$ and $p(v, u)$ now involves disjoint sets of feature functions. Thus, $p(u, v)$ is not guaranteed to satisfy the complementarity condition imposed on $h(u, v)$, and therefore cannot directly be used as the binary pairwise comparator. We resolve this by normalizing across the two possible permutations, as follows:

$$h'(u, v) = \frac{p(u, v)}{p(u, v) + p(v, u)} \qquad (3)$$

$$h'(v, u) = \frac{p(v, u)}{p(u, v) + p(v, u)} \qquad (4)$$

Since $h'(u, v) + h'(v, u) = 1$, the complementarity constraint is now satisfied, and $h(u, v)$ is just a binarized (thresholded) version of $h'(u, v)$. Thus, the binary pairwise comparator can be constructed from the permuted classifier outputs.

3.2 Training the Pairwise Comparator

Training the maximum-entropy classifier for the pairwise comparator requires a set of target labels and input feature functions, both of which are derived from the held-out development set $\mathbf{D}$. We begin by decoding the source sentences in $\mathbf{D}$ with the seed SMT system, followed by error analysis using the Translation Edit Rate (TER) measure (Snover et al., 2006). TER measures translation quality by computing the number of edits (insertions, substitutions, and deletions) and shifts required to transform a translation hypothesis to its corresponding reference. We then rank $\mathbf{D}$ in decreasing order of the number of post-shift edits, i.e. the number of insertions, substitutions, and deletions after the shift operation is completed. Since shifts are often due to word re-ordering issues within the SMT decoder (especially for phrase-based systems), we do not consider them as errors for the purpose of ranking $\mathbf{D}$. Sentences at the top of the ordered list $\mathbf{D}'$ contain the maximum number of translation errors.

For each pair of sentences $(u, v) : u < v$ in $\mathbf{D}'$, we generate two training entries. The first, signifying that u appears before v in $\mathbf{D}'$, assigns the label *true* to a trigger list consisting of standard feature functions derived from u, and complementary feature functions derived from v. The second, reinforcing this observation, assigns the label *false* to a trigger list consisting of complementary feature functions from u, and standard feature functions from v. The labeled training set (feature:label pairs) for the comparator can be expressed as follows:

$$\begin{aligned} \forall (u, v) \quad \in \quad & \mathbf{D}' : u < v, \\ & \{\mathbf{f}(u) \quad \mathbf{f}'(v)\} : true \\ & \{\mathbf{f}'(u) \quad \mathbf{f}(v)\} : false \end{aligned}$$

Thus, if there are d sentences in $\mathbf{D}'$, we obtain a total of $d(d - 1)$ labeled examples to train the comparator. We use the standard L-BFGS optimization algorithm (Liu and Nocedal, 1989) to estimate the parameters of the maximum entropy model.

3.3 Greedy Discriminative Selection

The discriminatively-trained pairwise comparator can be used as a relational operator to sort the candidate pool $\mathbf{P}$ in decreasing order of potential translation error reduction. A batch of pre-determined size K can then be selected from the top of this list to augment the existing SMT training corpus. Assuming the pool contains N candidate sentences, and given a fast sorting algorithm such as Quicksort, the complexity of this strategy is $O(N \log N)$. Batches can be selected iteratively until a specified performance threshold is achieved.

A potential downside of this approach reveals itself when there is redundancy in the candidate pool. Since the batch is selected in a single atomic operation from the sorted candidates, and because similar or identical sentences will typically occupy the same range in the ordered list, it is likely that this approach will result in batches with low diversity. Whereas we desire diverse batches for better coverage and efficient use of manual translation resources. This issue was previously addressed in Shen et al. (2004) in the context of named-entity recognition, where they used a two-step procedure to first select the most informative and representative samples, followed by a diversity filter. Ananthakrishnan et al. (2010) used a greedy, incremental batch construction strategy with an integrated, explicit batch diversity feature as part of the ranking model. Based on these ideas, we design a greedy selection strategy using the discriminative relational operator.

Rather than perform a full sort on $\mathbf{P}$, we simply invoke the $min_{h(u,v)}(\cdots)$ function to find the sentence that potentially minimizes translation error. The subscript indicates that our implementation of this function utilizes the discriminative relational operator trained on the development set $\mathbf{D}$. The best choice sentence s is then added to our batch at the current position (we begin with an empty batch). We then remove the standard and complementary feature functions $\mathbf{f}(s)$ and $\mathbf{f}'(s)$ triggered by s from the global pool of feature functions obtained from $\mathbf{D}$, so that they do not play a role in the selection of subsequent sentences for the batch. Subsequently, a candidate sentence that is similar or identical to s will not be preferred, because the feature functions that previously caused it to rank highly will no longer trigger. Algorithm 1 summarizes our selection strategy in pseudocode. Since each call to $min_{h(u,v)}(\cdots)$ is $O(N)$, the overall complexity of greedy discriminative selection is $O(K \cdot N)$.

Algorithm 1 Greedy Discriminative Selection

```
$\mathbf{B} \leftarrow ()$
for $k = 1$ to $K$ do
    $s \leftarrow min_{h(u,v)}(\mathbf{P})$
    $B(k) \leftarrow s$
    $\mathbf{P} \leftarrow \mathbf{P} - \{s\}$
    $\mathbf{f}(\mathbf{D}) \leftarrow \mathbf{f}(\mathbf{D}) - \mathbf{f}(s)$
    $\mathbf{f}'(\mathbf{D}) \leftarrow \mathbf{f}'(\mathbf{D}) - \mathbf{f}'(s)$
end for
return $\mathbf{B}$
```

4 Experiments and Results

We conduct a variety of simulation experiments with multiple language pairs (English-Pashto and Spanish-English) and different data configurations in order to demonstrate the utility of discriminative sample selection in the context of resource-poor SMT. We also compare the performance of the proposed strategy to numerous competing active and passive selection methods as follows:

- *Random*: Source sentences are uniformly sampled from the candidate pool $\mathbf{P}$.
- *Similarity*: Choose sentences from $\mathbf{P}$ with the highest fraction of n-gram overlap with the seed corpus $\mathbf{S}$.
- *Dissimilarity*: Select sentences from $\mathbf{P}$ with the highest proportion of n-grams not seen in the seed corpus $\mathbf{S}$ (Eck et al., 2005; Haffari et al., 2009).
- *Longest*: Pick the longest sentences from the candidate pool $\mathbf{P}$.
- *Semi-supervised*: Semi-supervised active learning with greedy incremental selection (Ananthakrishnan et al., 2010).
- *Discriminative*: Choose sentences that potentially minimize translation error using a maximum-entropy pairwise comparator (proposed method).

Identical low-resource initial conditions are applied to each selection strategy so that they may be objectively compared. A very small seed corpus $\mathbf{S}$ is sampled from the available parallel training data; the remainder serves as the candidate pool. Following the literature on active learning for SMT, our simulation experiments are iterative. A fixed-size batch of source sentences is constructed from the candidate pool using one of the above selection strategies. We then look up the corresponding translations from the candidate targets (simulating an expert human translator), augment the seed corpus with the selected data, and update the SMT system with the expanded training corpus. The selected data are removed from the candidate pool. This select-update cycle is then repeated for either a fixed number of iterations or until a specified performance benchmark is attained. At each iteration, we decode the unseen test set $\mathbf{T}$ with the most current SMT configuration and evaluate translation performance in terms of BLEU as well as coverage (defined as the fraction of untranslatable source words in the target hypotheses).

We use a phrase-based SMT framework similar to Koehn et al. (2003) for all experiments.

4.1 English-Pashto Simulation

Our English-Pashto (E2P) data originates from a two-way collection of spoken dialogues, and consists of two parallel sub-corpora: a directional E2P corpus and a directional Pashto-English (P2E) corpus. Each sub-corpus has its own independent training, development, and test partitions. The directional E2P training, development, and test sets consist of 33.9k, 2.4k, and 1.1k sentence pairs, respectively. The directional P2E training set consists of 76.5k sentence pairs. The corpus was used as-is, i.e. no length-based filtering or redundancy-reduction (i.e. removal of duplicates, if any) was performed. The test-set BLEU score with the baseline E2P SMT system trained from all of the above data was 9.5%.

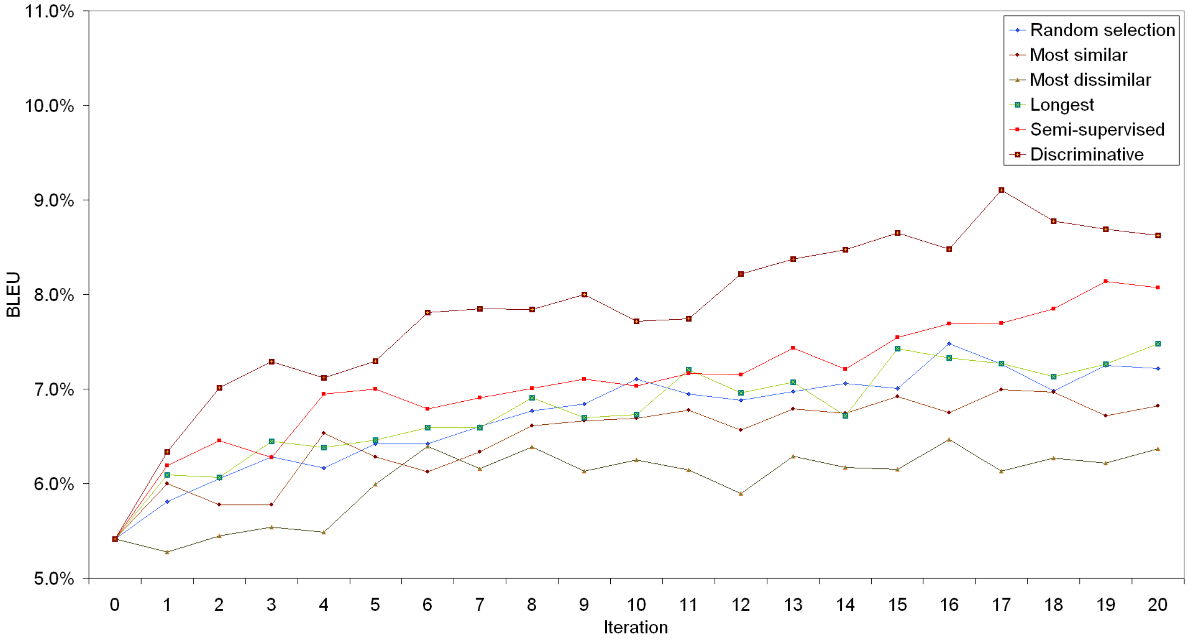
We obtained a seed training corpus by randomly sampling 1,000 sentence pairs from the directional E2P training partition. The remainder of this set, and the entire reversed P2E training partition were combined to create the pool (109.4k sentence pairs). In the past, we have observed that the reversed directional P2E data gives very little performance gain in the E2P direction even though its vocabulary is similar, and can be considered "out-of-domain" as far as the E2P translation task is concerned. Thus, our pool consists of 30% in-domain and 70% out-of-domain sentence pairs, making for a challenging active learning problem. A pool training set of 10k source sentences is sampled from this collection for the semi-supervised selection strategy, leaving us with 99.4k candidate sentences, which we use for all competing techniques. The data configuration used in this simulation is identical to Ananthakrishnan et al. (2010), allowing us to compare various strategies under the same conditions. We simulated a total of 20 iterations with batches of 200 sentences each; the original 1,000 sample seed corpus grows to 5,000 sentence pairs and the end of our simulation.

Figure 1(a) illustrates the variation in BLEU scores across iterations for each selection strategy. The proposed discriminative sample selection technique performs significantly better at every iteration than random, similarity, dissimilarity, longest, and semi-supervised active selection. At the end of 20 iterations, the BLEU score gained 3.21 points, a relative improvement of 59.3%. This was followed by semi-supervised active learning, which improved by 2.66 BLEU points, a 49.2% relative improvement. Table 2 summarizes the total number of words selected by each strategy, as well as the total area under the BLEU curve with respect to the baseline. The latter, labeled $\text{BLEU}_{\text{area}}$ and expressed in *percent-iterations*, is a better measure of the overall performance of each strategy across all iterations than comparing BLEU scores at the final iteration.

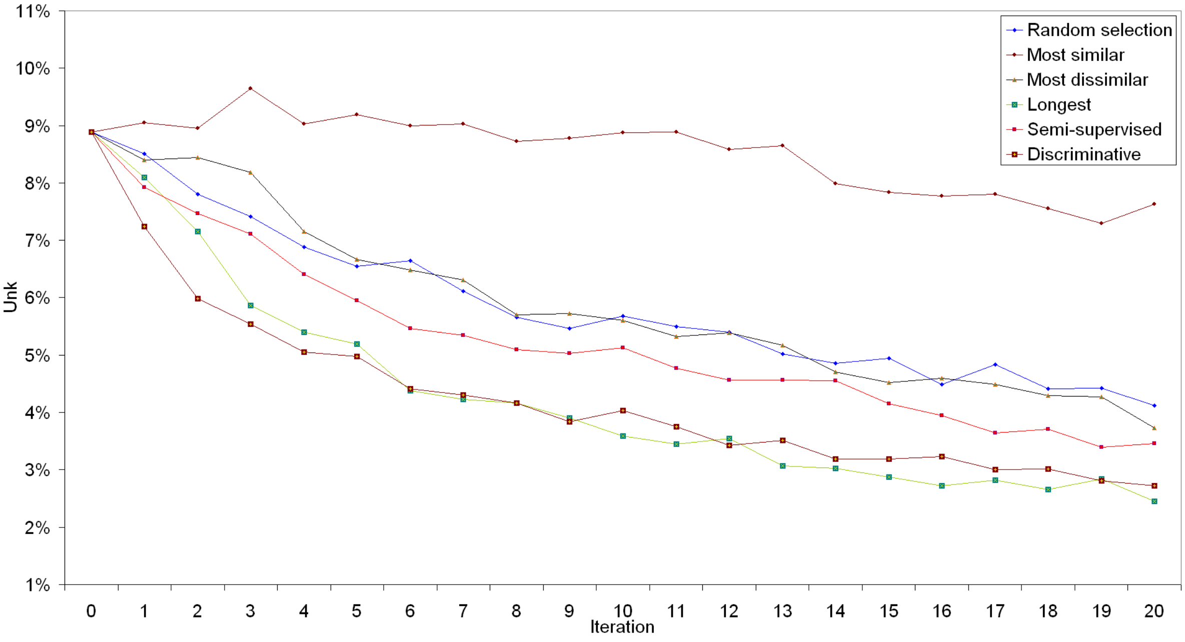
Figure 1(b) shows the variation in coverage (percentage of untranslatable source words in target hypotheses) for each selection technique. Here, discriminative sample selection was better than all other approaches except longest-sentence selection.

4.2 Spanish-English Simulation

The Spanish-English (S2E) training corpus was drawn from the Europarl collection (Koehn, 2005). To prevent length bias in selection, the corpus was filtered to only retain sentence pairs whose source ranged between 7 and 15 words (excluding punctuation). Additionally, redundancy was reduced by removing all duplicate sentence pairs. After these steps, we obtained approximately 253k sentence pairs for training. The WMT10 held-out develop-

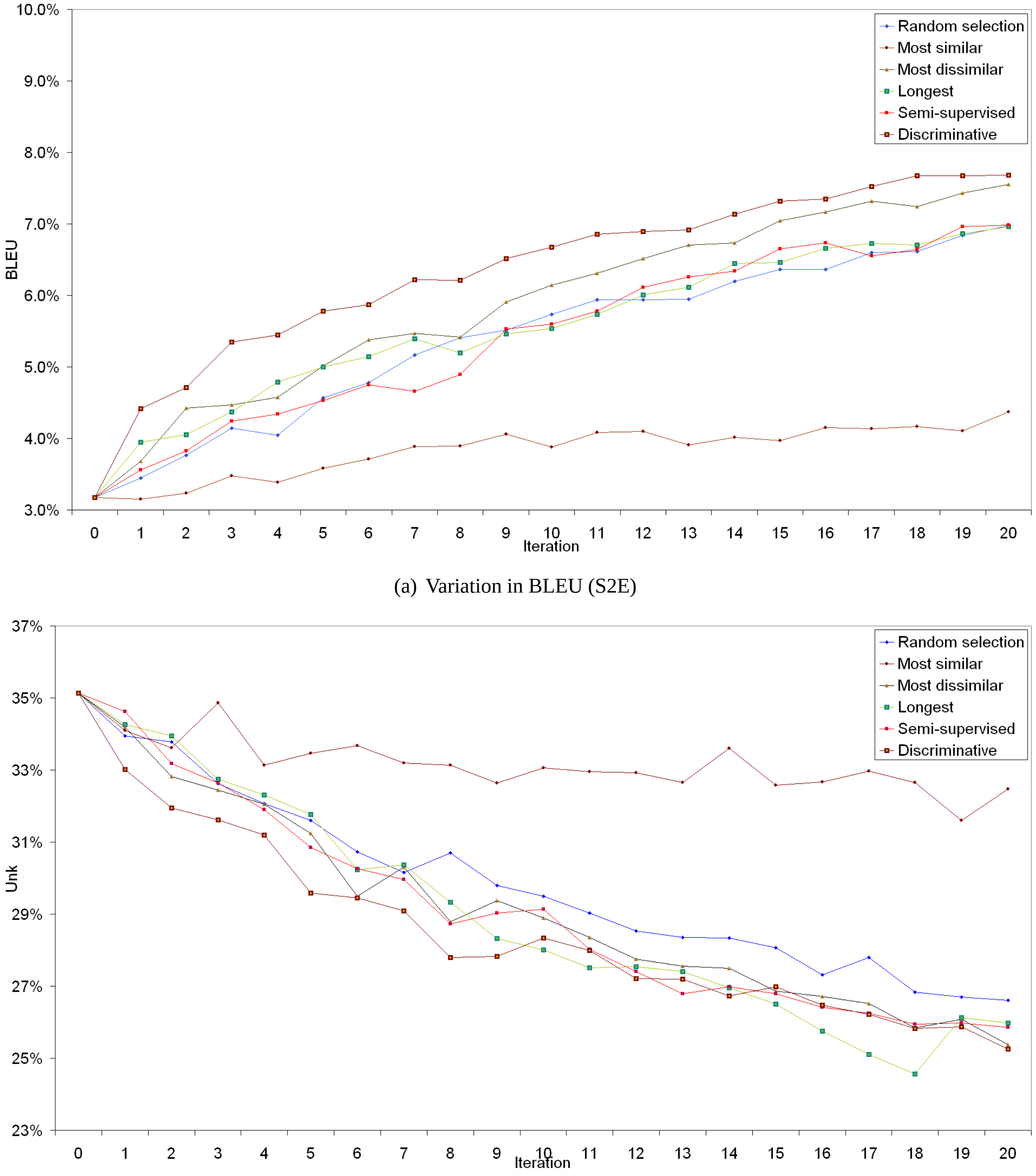


(a) Variation in BLEU (E2P)



(b) Variation in coverage (E2P)

Figure 1: Simulation results for E2P data selection.



(a) Variation in BLEU (S2E)

(b) Variation in coverage (S2E)

Figure 2: Simulation results for S2E data selection.

Method	**E2P size**	**E2P BLEU$_{\text{area}}$**	**S2E size**	**S2E BLEU$_{\text{area}}$**
Random	58.1k	26.4	26.5k	45.0
Similarity	30.7k	21.9	24.7k	13.2
Dissimilarity	39.2k	12.4	24.2k	54.9
Longest	173.0k	27.5	39.6k	48.3
Semi-supervised	80.0k	34.1	27.6k	45.6
Discriminative	109.1k	49.6	31.0k	64.5

Table 2: Source corpus size (in words) and BLEU$_{\text{area}}$ after 20 sample selection iterations.

ment and test sets (2k and 2.5k sentence pairs, respectively) were used to tune our system and evaluate performance. Note that this data configuration is different from that of the E2P simulation in that there is no logical separation of the training data into "in-domain" and "out-of-domain" sets. The baseline S2E SMT system trained with all available data gave a test-set BLEU score of 17.2%.

We randomly sampled 500 sentence pairs from the S2E training partition to obtain a seed training corpus. The remainder, after setting aside another 10k source sentences for training the semi-supervised strategy, serves as the candidate pool. We again simulated a total of 20 iterations, except in this case, we used batches of 100 sentences in an attempt to obtain smoother performance trajectories. The training corpus grows from 500 sentence pairs to 2,500 as the simulation progresses.

Variation in BLEU scores and coverage for the S2E simulation are illustrated in Figures 2(a) and 2(b), respectively. Discriminative sample selection outperformed all other selection techniques across all iterations of the simulation. After 20 iterations, we obtained a 4.51 point gain in BLEU, a relative improvement of 142.3%. The closest competitor was dissimilarity-based selection, which improved by 4.38 BLEU points, a 138.1% relative improvement. The proposed method also outperformed other selection strategies in improving coverage, with significantly better results especially in the early iterations. Table 2 summarizes the number of words chosen, and BLEU$_{\text{area}}$, for each strategy.

5 Conclusion and Future Directions

Building SMT systems for resource-poor language pairs requires significant investment of labor, time, and money for the development of parallel training corpora. We proposed a novel, discriminative sample selection strategy that can help lower these costs by choosing batches of source sentences from a large candidate pool. The chosen sentences, in conjunction with their manual translations, provide significantly better SMT performance than numerous competing active and passive selection techniques.

Our approach hinges on a maximum-entropy pairwise comparator that serves as a relational operator for comparing two source sentences. This allows us to rank the candidate pool in decreasing order of potential reduction in translation error with respect to an existing seed SMT system. The discriminative comparator is coupled with a greedy, incremental selection technique that discourages redundancy in the chosen batches. The proposed technique diverges from existing work on active sample selection for SMT in that it uses machine learning techniques in an attempt to explicitly reduce translation error by choosing sentences whose constituents were incorrectly translated in a held-out development set.

While the performance of competing strategies varied across language pairs and data configurations, discriminative sample selection proved consistently superior under all test conditions. It provides a powerful, flexible, data selection front-end for rapid development of SMT systems. Unlike some selection techniques, it is also platform-independent, and can be used as-is with a phrase-based, hierarchical, syntactic, or other SMT framework.

We have so far restricted our experiments to simulations, obtaining expert human translations directly from the sequestered parallel corpus. We are now actively exploring the possibility of linking the sample selection front-end to a crowd-sourcing backend, in order to obtain "non-expert" translations using a platform such as the Amazon Mechanical Turk.

References

Nir Ailon and Mehryar Mohri. 2008. An efficient reduction of ranking to classification. In *COLT '08: Proceedings of the 21st Annual Conference on Learning Theory*, pages 87–98.

Sankaranarayanan Ananthakrishnan, Rohit Prasad, David Stallard, and Prem Natarajan. 2010. A semi-supervised batch-mode active learning strategy for improved statistical machine translation. In *CoNLL '10: Proceedings of the 14th International Conference on Computational Natural Language Learning*, pages 126–134, July.

David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. 1996. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4(1):129–145.

Matthias Eck, Stephan Vogel, and Alex Waibel. 2005. Low cost portability for statistical machine translation based in N-gram frequency and TF-IDF. In *Proceedings of IWSLT*, Pittsburgh, PA, October.

Gholamreza Haffari, Maxim Roy, and Anoop Sarkar. 2009. Active learning for statistical phrase-based machine translation. In *NAACL '09: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 415–423, Morristown, NJ, USA. Association for Computational Linguistics.

Rebecca Hwa. 2004. Sample selection for statistical parsing. *Computational Linguistics*, 30:253–276.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54, Morristown, NJ, USA. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit X: Proceedings of the 10th Machine Translation Summit*, pages 79–86.

D. C. Liu and J. Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Math. Program.*, 45(3):503–528.

Qinggang Meng and Mark Lee. 2008. Error-driven active learning in growing radial basis function networks for early robot learning. *Neurocomputing*, 71(7-9):1449–1461.

Dan Shen, Jie Zhang, Jian Su, Guodong Zhou, and Chew-Lim Tan. 2004. Multi-criteria-based active learning for named entity recognition. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 589–596, Morristown, NJ, USA. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings AMTA*, pages 223–231, August.

Min Tang, Xiaoqiang Luo, and Salim Roukos. 2002. Active learning for statistical natural language parsing. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 120–127, Morristown, NJ, USA. Association for Computational Linguistics.

Effects of Empty Categories on Machine Translation

Tagyoung Chung and **Daniel Gildea**
Department of Computer Science
University of Rochester
Rochester, NY 14627

Abstract

We examine effects that empty categories have on machine translation. Empty categories are elements in parse trees that lack corresponding overt surface forms (words) such as dropped pronouns and markers for control constructions. We start by training machine translation systems with manually inserted empty elements. We find that inclusion of some empty categories in training data improves the translation result. We expand the experiment by automatically inserting these elements into a larger data set using various methods and training on the modified corpus. We show that even when automatic prediction of null elements is not highly accurate, it nevertheless improves the end translation result.

1 Introduction

An empty category is an element in a parse tree that does not have a corresponding surface word. They include traces such as Wh-traces which indicate movement operations in interrogative sentences and dropped pronouns which indicate omission of pronouns in places where pronouns are normally expected. Many treebanks include empty nodes in parse trees to represent non-local dependencies or dropped elements. Examples of the former include traces such as relative clause markers in the Penn Treebank (Bies et al., 1995). An example of the latter include dropped pronouns in the Korean Treebank (Han and Ryu, 2005) and the Chinese Treebank (Xue and Xia, 2000).

In languages such as Chinese, Japanese, and Korean, pronouns are frequently or regularly dropped when they are pragmatically inferable. These languages are called *pro-drop* languages. Dropped pronouns are quite a common phenomenon in these languages. In the Chinese Treebank, they occur once in every four sentences on average. In Korean the Treebank, they are even more frequent, occurring in almost every sentence on average. Translating these pro-drop languages into languages such as English where pronouns are regularly retained could be problematic because English pronouns have to be generated from nothing.

There are several different strategies to counter this problem. A special NULL word is typically used when learning word alignment (Brown et al., 1993). Words that have non-existent counterparts can be aligned to the NULL word. In phrase-based translation, the phrase learning system may be able to learn pronouns as a part of larger phrases. If the learned phrases include pronouns on the target side that are dropped from source side, the system may be able to insert pronouns even when they are missing from the source language. This is an often observed phenomenon in phrase-based translation systems. Explicit insertion of missing words can also be included in syntax-based translation models (Yamada and Knight, 2001). For the closely related problem of inserting grammatical function particles in English-to-Korean and English-to-Japanese machine translation, Hong et al. (2009) and Isozaki et al. (2010) employ preprocessing techniques to add special symbols to the English source text.

In this paper, we examine a strategy of automatically inserting two types of empty elements from the Korean and Chinese treebanks as a preprocess-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 636–645,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

Korean		
T	0.47	trace of movement
(NP *pro*)	0.88	dropped subject or object
(WHNP *op*)	0.40	empty operator in relative constructions
?	0.006	verb deletion, VP ellipsis, and others

Chinese		
(XP (-NONE- *T*))	0.54	trace of A'-movement
(NP (-NONE- *))	0.003	trace of A-movement
(NP (-NONE- *pro*))	0.27	dropped subject or object
(NP (-NONE- *PRO*))	0.31	control structures
(WHNP (-NONE- *OP*))	0.53	empty operator in relative constructions
(XP (-NONE- *RNR*))	0.026	right node raising
(XP (-NONE- *?*))	0	others

Table 1: List of empty categories in the Korean Treebank (top) and the Chinese Treebank (bottom) and their per-sentence frequencies in the training data of initial experiments.

		BLEU
Chi-Eng	No null elements	19.31
	w/ *pro*	19.68
	w/ *PRO*	19.54
	w/ *pro* and *PRO*	20.20
	w/ all null elements	**20.48**
Kor-Eng	No null elements	20.10
	w/ *pro*	**20.37**
	w/ all null elements	19.71

Table 2: BLEU score result of initial experiments. Each experiment has different empty categories added in. *PRO* stands for the empty category used to mark control structures and *pro* indicates dropped pronouns for both Chinese and Korean.

ing step. We first describe our experiments with data that have been annotated with empty categories, focusing on zero pronouns and traces such as those used in control constructions. We use these annotations to insert empty elements in a corpus and train a machine translation system to see if they improve translation results. Then, we illustrate different methods we have devised to automatically insert empty elements to corpus. Finally, we describe our experiments with training machine translation systems with corpora that are automatically augmented with empty elements. We conclude this paper by discussing possible improvements to the different methods we describe in this paper.

2 Initial experiments

2.1 Setup

We start by testing the plausibility of our idea of preprocessing corpus to insert empty categories with ideal datasets. The Chinese Treebank (LDC2005T01U01) is annotated with null elements and a portion of the Chinese Treebank has been translated into English (LDC2007T02). The Korean Treebank version 1.0 (LDC2002T26) is also annotated with null elements and includes an English translation. We extract null elements along with tree terminals (words) and train a simple phrase-based machine translation system. Both datasets have about 5K sentences and 80% of the data was used for training, 10% for development, and 10% for testing.

We used Moses (Koehn et al., 2007) to train machine translation systems. Default parameters were used for all experiments. The same number of GIZA++ (Och and Ney, 2003) iterations were used for all experiments. Minimum error rate training (Och, 2003) was run on each system afterwards, and the BLEU score (Papineni et al., 2002) was calculated on the test sets.

There are several different empty categories in the different treebanks. We have experimented with leaving in and out different empty categories for different experiments to see their effect. We hypothesized that nominal phrasal empty categories such as dropped pronouns may be more useful than other ones, since they are the ones that may be missing in the source language (Chinese and Korean) but have counterparts in the target (English). Table 1 summarizes empty categories in Chinese and Korean treebank and their frequencies in the training data.

2.2 Results

Table 2 summarizes our findings. It is clear that not all elements improve translation results when included in the training data. For the Chinese to English experiment, empty categories that mark control structures (*PRO*), which serve as the subject of a dependent clause, and dropped pronouns (*pro*), which mark omission of pragmatically in-

word	$P(e \mid *\text{pro}*)$	word	$P(e \mid *\text{PRO}*)$
the	0.18	to	0.45
i	0.13	NULL	0.10
it	0.08	the	0.02
to	0.08	of	0.02
they	0.05	as	0.02

Table 3: A lexical translation table from the Korean-English translation system (left) and a lexical translation from the Chinese-English translation system (right). For the Korean-English lexical translation table, the left column is English words that are aligned to a dropped pronoun (*pro*) and the right column is the conditional probability of $P(e \mid *\text{pro}*)$. For the Chinese-English lexical translation table, the left column is English words that are aligned to a control construction marker (*PRO*) and the right column is the conditional probability of $P(e \mid *\text{PRO}*)$.

ferable pronouns, helped to improve translation results the most. For the Korean to English experiment, the dropped pronoun is the only empty category that seems to improve translation.

For the Korean to English experiment, we also tried annotating whether the dropped pronouns are a subject, an object, or a complement using information from the Treebank's function tags, since English pronouns are inflected according to case. However, this did not yield a very different result and in fact was slightly worse. This is possibly due to data sparsity created when dropped pronouns are annotated. Dropped pronouns in subject position were the overwhelming majority (91%), and there were too few dropped pronouns in object position to learn good parameters.

2.3 Analysis

Table 3 and Table 4 give us a glimpse of why having these empty categories may lead to better translation. Table 3 is the lexical translation table for the dropped pronoun (*pro*) from the Korean to English experiment and the marker for control constructions (*PRO*) from the Chinese to English experiment. For the dropped pronoun in the Korean to English experiment, although there are errors, the table largely reflects expected translations of a dropped pronoun. It is possible that the system is inserting pronouns in right places that would be missing otherwise. For the control construction marker in the Chinese to English experiment, the top translation for *PRO* is the English word *to*, which is expected since Chinese clauses that have control construction markers often translate to English as to-infinitives. However, as we discuss in the next paragraph, the presence of control construction markers may affect translation results in more subtle ways when combined with phrase learning.

Table 4 shows how translations from the system trained with null elements and the system trained without null elements differ. The results are taken from the test set and show extracts from larger sentences. Chinese verbs that follow the empty node for control constructions (*PRO*) are generally translated to English as a verb in to-infinitive form, a gerund, or a nominalized verb. The translation results show that the system trained with this null element (*PRO*) translates verbs that follow the null element largely in such a manner. However, it may not be always closest to the reference. It is exemplified by the translation of one phrase.

Experiments in this section showed that preprocessing the corpus to include some empty elements can improve translation results. We also identified which empty categories maybe helpful for improving translation for different language pairs. In the next section, we focus on how we add these elements automatically to a corpus that is not annotated with empty elements for the purpose of preprocessing corpus for machine translation.

3 Recovering empty nodes

There are a few previous works that have attempted restore empty nodes for parse trees using the Penn English Treebank. Johnson (2002) uses rather simple pattern matching to restore empty categories as well as their co-indexed antecedents with surprisingly good accuracy. Gabbard et al. (2006) present a more sophisticated algorithm that tries to recover empty categories in several steps. In each step, one or more empty categories are restored using patterns or classifiers (five maximum-entropy and two perceptron-based classifiers to be exact).

What we are trying to achieve has obvious similarity to these previous works. However, there are several differences. First, we deal with different languages. Second, we are only trying to recover

Chinese	English Reference	System trained w/ nulls	System trained w/o nulls
PRO 贯彻	implementing	implementation	implemented
PRO 逐步 形成	have gradually formed	to gradually form	gradually formed
PRO 吸引 外资 作为	attracting foreign investment	attracting foreign investment	attract foreign capital

Table 4: The first column is a Chinese word or a phrase that immediately follows empty node marker for Chinese control constructions. The second column is the English reference translation. The third column is the translation output from the system that is trained with the empty categories added in. The fourth column is the translation output from the system trained without the empty categories added, which was given the test set without the empty categories. Words or phrases and their translations presented in the table are part of larger sentences.

a couple of empty categories that would help machine translation. Third, we are not interested in recovering antecedents. The linguistic differences and the empty categories we are interested in recovering made the task much harder than it is for English. We will discuss this in more detail later.

From this section on, we will discuss only Chinese-English translation because Chinese presents a much more interesting case, since we need to recover two different empty categories that are very similarly distributed. Data availability was also a consideration since much larger datasets (bilingual and monolingual) are available for Chinese. The Korean Treebank has only about 5K sentences, whereas the version of Chinese Treebank we used includes 28K sentences.

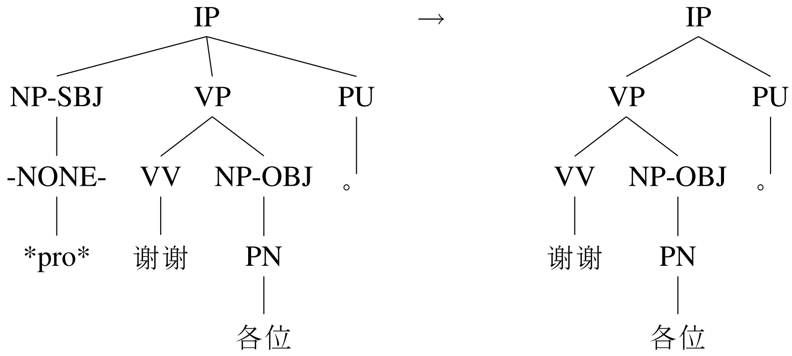
The Chinese Treebank was used for all experiments that are mentioned in the rest of this Section. Roughly 90% of the data was used for the training set, and the rest was used for the test set. As we have discussed in Section 2, we are interested in recovering dropped pronouns (*pro*) and control construction markers (*PRO*). We have tried three different relatively simple methods so that recovering empty elements would not require any special infrastructure.

3.1 Pattern matching

Johnson (2002) defines a pattern for empty node recovery to be a minimally connected tree fragment containing an empty node and all nodes co-indexed with it. Figure 1 shows an example of a pattern. We extracted patterns according this definition, and it became immediately clear that the same definition that worked for English will not work for Chinese. Table 5 shows the top five patterns that match control constructions (*PRO*) and dropped pronouns (*pro*). The top pattern that matches *pro* and *PRO* are both exactly the same, since the pattern will be matched against parse trees where empty nodes have been deleted.

When it became apparent that we cannot use the same definition of patterns to successfully restore empty categories, we added more context to the patterns. Patterns needed more context for them to be able to disambiguate between sites that need to be inserted with *pro*s and sites that need to be inserted with *PRO*s. Instead of using minimal tree fragments that matched empty categories, we included the parent and siblings of the minimal tree fragment in the pattern (pattern matching method 1). This way, we gained more context. However, as can be seen in Table 5, there is still a lot of overlap between patterns for the two empty categories. However, it is more apparent that at least we can choose the pattern that will maximize matches for one empty category and then discard that pattern for the other empty category.

We also tried giving patterns even more context by including terminals if preterminals are present in the pattern (pattern matching method 2). In this way, we are able have more context for patterns such as (VP VV (IP (NP (-NONE- *PRO*)) VP)) by knowing what the verb that precedes the empty category is. Instead of the original pattern, we would have patterns such as (VP (VV 决 定) (IP (NP (-NONE- *PRO*)) VP)). We are able to gain more context because some verbs select for a control construction. The Chinese verb 决定 generally translates to English as *to decide* and is more often followed by a control construction than by a dropped pronoun. Whereas the pattern (VP (VV 决定) (IP (NP (-NONE- *PRO*)) VP)) occurred 154 times in the training data, the pattern (VP (VV 决定) (IP (NP (-NONE- *pro*)) VP)) occurred only 8 times in the



(IP (NP-SBJ (-NONE- *pro*)) VP PU) (IP VP PU)

Figure 1: An example of a tree with an empty node (left), the tree stripped of an empty node (right), and a pattern that matches the example. Sentences are parsed without empty nodes and if a tree fragment (IP VP PU) is encountered in a parse tree, the empty node may be inserted according to the learned pattern (IP (NP-SBJ (-NONE- *pro*)) VP PU).

	PRO		*pro*
Count	Pattern	Count	Pattern
12269	(IP (NP (-NONE- *PRO*)) VP)	10073	(IP (NP (-NONE- *pro*)) VP)
102	(IP PU (NP (-NONE- *PRO*)) VP PU)	657	(IP (NP (-NONE- *pro*)) VP PU)
14	(IP (NP (-NONE- *PRO*)) VP PRN)	415	(IP ADVP (NP (-NONE- *pro*)) VP)
13	(IP NP (NP (-NONE- *PRO*)) VP)	322	(IP NP (NP (-NONE- *pro*)) VP)
12	(CP (NP (-NONE- *PRO*)) CP)	164	(IP PP PU (NP (-NONE- *pro*)) VP)

	PRO		*pro*
Count	Pattern	Count	Pattern
2991	(VP VV NP (IP (NP (-NONE- *PRO*)) VP))	1782	(CP (IP (NP (-NONE- *pro*)) VP) DEC)
2955	(VP VV (IP (NP (-NONE- *PRO*)) VP))	1007	(VP VV (IP (NP (-NONE- *pro*)) VP))
850	(CP (IP (NP (-NONE- *PRO*)) VP) DEC)	702	(LCP (IP (NP (-NONE- *pro*)) VP) LC)
765	(PP P (IP (NP (-NONE- *PRO*)) VP))	684	(IP IP PU (IP (NP (-NONE- *pro*)) VP) PU)
654	(LCP (IP (NP (-NONE- *PRO*)) VP) LC)	654	(TOP (IP (NP (-NONE- *pro*)) VP PU))

Table 5: Top five minimally connected patterns that match *pro* and *PRO* (top). Patterns that match both *pro* and *PRO* are shaded with the same color. The table on the bottom show more refined patterns that are given added context by including the parent and siblings to minimally connected patterns. Many patterns still match both *pro* and *PRO* but there is a lesser degree of overlap.

training data.

After the patterns are extracted, we performed pruning similar to the pruning that was done by Johnson (2002). The patterns that have less than 50% chance of matching are discarded. For example, if (IP VP) occurs one hundred times in a treebank that is stripped of empty nodes and if pattern (IP (NP (-NONE- *PRO*)) VP) occurs less than fifty times in the same treebank that is annotated with empty nodes, it is discarded.[1] We also found that we can discard patterns that occur very rarely (that occur only once) without losing much accuracy. In cases where there was an overlap between two empty categories, the pattern was chosen for either *pro* or *PRO*, whichever that maximized the number of matchings and then discarded for the other.

3.2 Conditional random field

We tried building a simple conditional random field (Lafferty et al., 2001) to predict null elements. The model examines each and every word boundary and decides whether to leave it as it is, insert *pro*, or insert *PRO*. The obvious disadvantage of this method is that if there are two consecutive null elements, it will miss at least one of them. Although there were some cases like this in the treebank, they were rare enough that we decided to ignore them. We first tried using only differently sized local windows of words as features (CRF model 1). We also experimented with adding the part-of-speech tags of words as features (CRF model 2). Finally, we experimented with a variation where the model is given each word and its part-of-speech tag and its immediate parent node as features (CRF model 3).

We experimented with using different regularizations and different values for regularizations but it did not make much difference in the final results. The numbers we report later used L_2 regularization.

3.3 Parsing

In this approach, we annotated nonterminal symbols in the treebank to include information about empty categories and then extracted a context free grammar from the modified treebank. We parsed with the modified grammar, and then deterministically recovered the empty categories from the trees. Figure 2 illustrates how the trees were modified. For every empty node, the most immediate ancestor of the empty node that has more than one child was annotated with information about the empty node, and the empty node was deleted. We annotated whether the deleted empty node was *pro* or *PRO* and where it was deleted. Adding where the child was necessary because, even though most empty nodes are the first child, there are many exceptions.

	PRO			*pro*		
Cycle	Prec.	Rec.	F1	Prec	Rec.	F1
1	0.38	0.08	0.13	0.38	0.08	0.12
2	0.52	0.23	0.31	0.37	0.18	0.24
3	0.59	0.46	0.52	0.43	0.24	0.31
4	**0.62**	0.50	0.56	**0.47**	0.25	0.33
5	0.61	0.52	**0.56**	0.47	0.33	0.39
6	0.60	**0.53**	0.56	0.46	0.39	**0.42**
7	0.58	0.52	0.55	0.43	**0.40**	0.41

Table 6: Result using the grammars output by the Berkeley state-splitting grammar trainer to predict empty categories

We first extracted a plain context free grammar after modifying the trees and used the modified grammar to parse the test set and then tried to recover the empty elements. This approach did not work well. We then applied the latent annotation learning procedures of Petrov et al. (2006)[2] to refine the nonterminals in the modified grammar. This has been shown to help parsing in many different situations. Although the state splitting procedure is designed to maximize the likelihood of of the parse trees, rather than specifically to predict the empty nodes, learning a refined grammar over modified trees was also effective in helping to predict empty nodes. Table 6 shows the dramatic improvement after each split, merge, and smoothing cycle. The gains leveled off after the sixth iteration and the sixth order grammar was used to run later experiments.

3.4 Results

Table 7 shows the results of our experiments. The numbers are very low when compared to accuracy reported in other works that were mentioned in the beginning of this Section, which dealt with the Penn English Treebank. Dropped pronouns are especially

[1] See Johnson (2002) for more details.

[2] http://code.google.com/p/berkeleyparser/

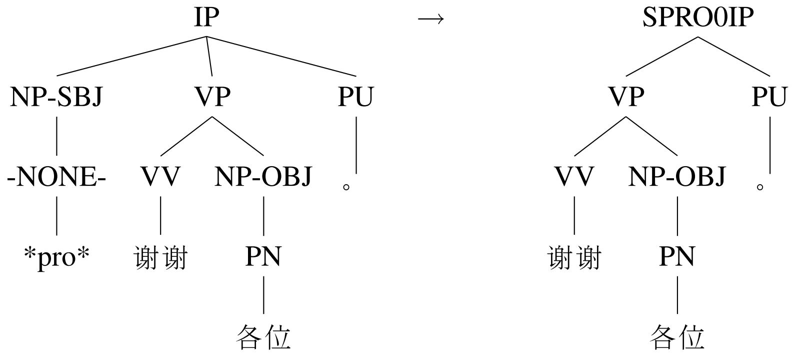


Figure 2: An example of tree modification

	PRO			*pro*		
	Prec.	Rec.	F1	Prec	Rec.	F1
Pattern 1	0.65	**0.61**	**0.63**	0.41	0.23	0.29
Pattern 2	0.67	0.58	0.62	0.46	0.24	0.31
CRF 1	0.66	0.31	0.43	0.53	0.24	0.33
CRF 2	**0.68**	0.46	0.55	**0.58**	0.35	**0.44**
CRF 3	0.63	0.47	0.54	0.54	0.36	0.43
Parsing	0.60	0.53	0.56	0.46	**0.39**	0.42

Table 7: Result of recovering empty nodes

hard to recover. However, we are dealing with a different language and different kinds of empty categories. Empty categories recovered this way may still help translation. In the next section, we take the best variation of the each method use it to add empty categories to a training corpus and train machine translation systems to see whether having empty categories can help improve translation in more realistic situations.

3.5 Analysis

The results reveal many interesting aspects about recovering empty categories. The results suggest that tree structures are important features for finding sites where markers for control constructions (*PRO*) have been deleted. The method utilizing patterns that have more information about tree structure of these sites performed better than other methods. The fact that the method using parsing was better at predicting *PRO*s than the methods that used the conditional random fields also corroborates this finding. For predicting dropped pronouns, the method using the CRFs did better than the others. This suggests that rather than tree structure, local context of words and part-of-speech tags maybe more important features for predicting dropped pronouns. It may also suggest that methods using robust machine learning techniques are better outfitted for predicting dropped pronouns.

It is interesting to note how effective the parser was at predicting empty categories. The method using the parser requires the least amount of supervision. The method using CRFs requires feature design, and the method that uses patterns needs human decisions on what the patterns should be and pruning criteria. There is also room for improvement. The split-merge cycles learn grammars that produce better parse trees rather than grammars that predict empty categories more accurately. By modifying this learning process, we may be able to learn grammars that are better suited for predicting empty categories.

4 Experiments

4.1 Setup

For Chinese-English, we used a subset of FBIS newswire data consisting of about 2M words and 60K sentences on the English side. For our development set and test set, we had about 1000 sentences each with 10 reference translations taken from the NIST 2002 MT evaluation. All Chinese data was re-segmented with the CRF-based Stanford Chinese segmenter (Chang et al., 2008) that is trained on the segmentation of the Chinese Treebank for consistency. The parser used in Section 3 was used to parse the training data so that null elements could be recovered from the trees. The same method for recovering null elements was applied to the train-

	BLEU	BP	*PRO*	*pro*
Baseline	23.73	1.000		
Pattern	23.99	0.998	0.62	0.31
CRF	**24.69***	1.000	0.55	0.44
Parsing	23.99	1.000	0.56	0.42

Table 8: Final BLEU score result. The asterisk indicates statistical significance at $p < 0.05$ with 1000 iterations of paired bootstrap resampling. BP stands for the brevity penalty in BLEU. F1 scores for recovering empty categories are repeated here for comparison.

ing, development, and test sets to insert empty nodes for each experiment. The baseline system was also trained using the raw data.

We used Moses (Koehn et al., 2007) to train machine translation systems. Default parameters were used for all experiments. The same number of GIZA++ (Och and Ney, 2003) iterations were used for all experiments. Minimum error rate training (Och, 2003) was run on each system afterwards and the BLEU score (Papineni et al., 2002) was calculated on the test set.

4.2 Results

Table 8 summarizes our results. Generally, all systems produced BLEU scores that are better than the baseline, but the best BLEU score came from the system that used the CRF for null element insertion. The machine translation system that used training data from the method that was overall the best in predicting empty elements performed the best. The improvement is 0.96 points in BLEU score, which represents statistical significance at $p < 0.002$ based on 1000 iterations of paired bootstrap resampling (Koehn, 2004). Brevity penalties applied for calculating BLEU scores are presented to demonstrate that the baseline system is not penalized for producing shorter sentences compared other systems.[3]

The BLEU scores presented in Table 8 represent the best variations of each method we have tried for recovering empty elements. Although the difference was small, when the F1 score were same for two variations of a method, it seemed that we could get slightly better BLEU score with the variation that had higher recall for recovering empty elements rather the variation with higher precision.

We tried a variation of the experiment where the CRF method is used to recover *pro* and the pattern matching is used to recover *PRO*, since these represent the best methods for recovering the respective empty categories. However, it was not as successful as we thought would be. The resulting BLEU score from the experiment was 24.24, which is lower than the one that used the CRF method to recover both *pro* and *PRO*. The problem was we used a very naïve method of resolving conflict between two different methods. The CRF method identified 17463 sites in the training data where *pro* should be added. Of these sites, the pattern matching method guessed 2695 sites should be inserted with *PRO* rather than *pro*, which represent more than 15% of total sites that the CRF method decided to insert *pro*. In the aforementioned experiment, wherever there was a conflict, both *pro* and *PRO* were inserted. This probably lead the experiment to have worse result than using only the one best method. This experiment suggest that more sophisticated methods should be considered when resolving conflicts created by using heterogeneous methods to recover different empty categories.

Table 9 shows five example translations of source sentences in the test set that have one of the empty categories. Since empty categories have been automatically inserted, they are not always in the correct places. The table includes the translation results from the baseline system where the training and test sets did not have empty categories and the translation results from the system (the one that used the CRF) that is trained on an automatically augmented corpus and given the automatically augmented test set.

5 Conclusion

In this paper, we have showed that adding some empty elements can help building machine translation systems. We showed that we can still benefit from augmenting the training corpus with empty elements even when empty element prediction is less than what would be conventionally considered robust.

We have also shown that there is a lot of room for improvement. More comprehensive and sophisti-

[3] We thank an anonymous reviewer for tipping us to examine the brevity penalty.

source	中国 计划 *PRO* 投资 在 基础 设施 上
reference	china plans to invest in the infrastructure
system trained w/ nulls	china plans to invest in infrastructure
system trained w/o nulls	china 's investment in infrastructure
source	有利 *PRO* 巩固 香港 的 贸易 和 航运 中心
reference	good for consolidating the trade and shipping center of hong kong
system trained w/ nulls	favorable to the consolidation of the trade and shipping center in hong kong
system trained w/o nulls	hong kong will consolidate the trade and shipping center
source	一些 大型 企业 *PRO* 逐步 走向 破产
reference	some large - sized enterprises to gradually go bankrupt
system trained w/ nulls	some large enterprises to gradually becoming bankrupt
system trained w/o nulls	some large enterprises gradually becoming bankrupt
source	*pro* 目前 还 不 清楚
reference	it is not clear now
system trained w/ nulls	it is also not clear
system trained w/o nulls	he is not clear
source	*pro* 现在 还 不 清楚
reference	it is not clear yet
system trained w/ nulls	it is still not clear
system trained w/o nulls	is still not clear

Table 9: Sample translations. The system trained without nulls is the baseline system where the training corpus and test corpus did not have empty categories. The system trained with nulls is the system trained with the training corpus and the test corpus that have been automatically augmented with empty categories. All examples are part of longer sentences.

cated methods, perhaps resembling the work of Gabbard et al. (2006) may be necessary for more accurate recovery of empty elements. We can also consider simpler methods where different algorithms are used for recovering different empty elements, in which case, we need to be careful about how recovering different empty elements could interact with each other as exemplified by our discussion of the pattern matching algorithm in Section 3 and our experiment presented in Section 4.2.

There are several other issues we may consider when recovering empty categories that are missing in the target language. We only considered empty categories that are present in treebanks. However, there might be some empty elements which are not annotated but nevertheless helpful for improving machine translation. As always, preprocessing the corpus to address a certain problem in machine translation is less principled than tackling the problem head on by integrating it into the machine translation system itself. It may be beneficial to include consideration for empty elements in the decoding process, so that it can benefit from interacting with other elements of the machine translation system.

Acknowledgments We thank the anonymous reviewers for their helpful comments. This work was supported by NSF grants IIS-0546554 and IIS-0910611.

References

Ann Bies, Mark Ferguson, Karen Katz, and Robert MacIntyre. 1995. Bracketing guidelines for treebank II style. Penn Treebank Project, January.

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.

Pi-Chuan Chang, Michel Galley, and Christopher Manning. 2008. Optimizing Chinese word segmentation for machine translation performance. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 224–232.

Ryan Gabbard, Seth Kulick, and Mitchell Marcus. 2006. Fully parsing the Penn Treebank. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 184–191, New York

City, USA, June. Association for Computational Linguistics.

Na-Rae Han and Shijong Ryu. 2005. Guidelines for Penn Korean Treebank version 2.0. Technical report, IRCS, University of Pennsylvania.

Gumwon Hong, Seung-Wook Lee, and Hae-Chang Rim. 2009. Bridging morpho-syntactic gap between source and target sentences for English-Korean statistical machine translation. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 233–236.

Hideki Isozaki, Katsuhito Sudoh, Hajime Tsukada, and Kevin Duh. 2010. Head finalization: A simple reordering rule for sov languages. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and Metrics*, pages 244–251.

Mark Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the 40th Annual Conference of the Association for Computational Linguistics (ACL-02)*, Philadelphia, PA.

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. 2004. Statistical significance tests for machine translation evaluation. In *2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 388–395, Barcelona, Spain, July.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Machine Learning: Proceedings of the Eighteenth International Conference (ICML 2001)*, Stanford, California.

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

Franz Josef Och. 2003. Minimum error rate training for statistical machine translation. In *Proceedings of the 41th Annual Conference of the Association for Computational Linguistics (ACL-03)*.

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 Conference of the Association for Computational Linguistics (ACL-02)*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia, July. Association for Computational Linguistics.

Nianwen Xue and Fei Xia. 2000. The bracketing guidelines for the Penn Chinese Treebank. Technical Report IRCS-00-08, IRCS, University of Pennsylvania.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Conference of the Association for Computational Linguistics (ACL-01)*, Toulouse, France.

SCFG Decoding Without Binarization

Mark Hopkins and Greg Langmead
SDL Language Weaver, Inc.
6060 Center Drive, Suite 150
Los Angeles, CA 90045
{mhopkins,glangmead}@languageweaver.com

Abstract

Conventional wisdom dictates that synchronous context-free grammars (SCFGs) must be converted to Chomsky Normal Form (CNF) to ensure cubic time decoding. For arbitrary SCFGs, this is typically accomplished via the synchronous binarization technique of (Zhang et al., 2006). A drawback to this approach is that it inflates the constant factors associated with decoding, and thus the practical running time. (DeNero et al., 2009) tackle this problem by defining a superset of CNF called Lexical Normal Form (LNF), which also supports cubic time decoding under certain implicit assumptions. In this paper, we make these assumptions explicit, and in doing so, show that LNF can be further expanded to a broader class of grammars (called "scope-3") that also supports cubic-time decoding. By simply pruning non-scope-3 rules from a GHKM-extracted grammar, we obtain better translation performance than synchronous binarization.

1 Introduction

At the heart of bottom-up chart parsing (Younger, 1967) is the following combinatorial problem. We have a context-free grammar (CFG) rule (for instance, S → NP VP PP) and an input sentence of length n (for instance, "on the fast jet ski of mr smith"). During chart parsing, we need to apply the rule to all relevant subspans of the input sentence. See Figure 1. For this particular rule, there are $\binom{n+1}{4}$ *application contexts*, i.e. ways to choose the subspans. Since the asymptotic running time of chart parsing is at least linear in this quantity, it will take at least $O(\binom{n+1}{4}) = O(n^4)$ time if we include this rule in our grammar.

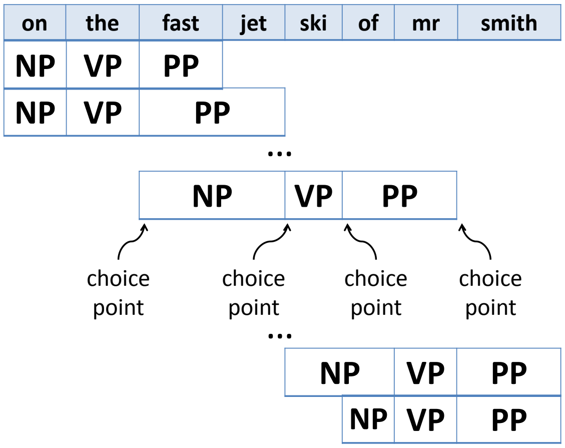


Figure 1: A demonstration of application contexts. There are $\binom{n+1}{4}$ application contexts for the CFG rule "S → NP VP PP", where n is the length of the input sentence.

Fortunately, we can take advantage of the fact that any CFG has an equivalent representation in Chomsky Normal Form (CNF). In CNF, all rules have the form X → Y Z or X → x, where x is a terminal and X, Y, Z are nonterminals. If a rule has the form X → Y Z, then there are only $\binom{n+1}{3}$ application contexts, thus the running time of chart parsing is $O(\binom{n+1}{3}) = O(n^3)$ when applied to CNF grammars.

A disadvantage to CNF conversion is that it increases both the overall number of rules and the overall number of nonterminals. This inflation of the "grammar constant" does not affect the asymptotic runtime, but can have a significant impact on the performance in practice. For this reason, (DeN-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 646–655,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

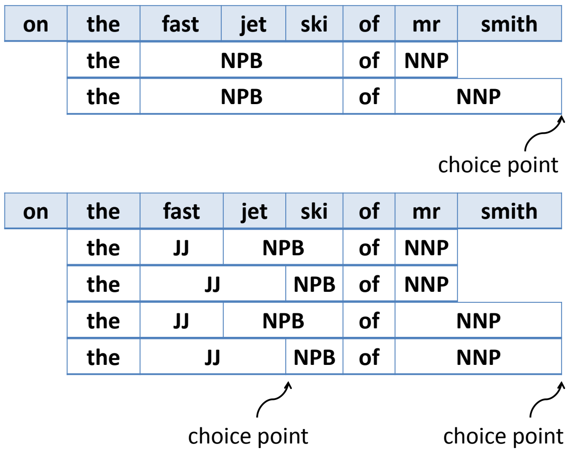


Figure 2: A demonstration of application contexts for rules with lexical anchors. There are $O(n)$ application contexts for CFG rule “S → the NPB of NNP”, and $O(n^2)$ application contexts for CFG rule “S → the JJ NPB of NNP”, if we assume that the input sentence has length n and contains no repeated words.

ero et al., 2009) provide a relaxation of CNF called Lexical Normal Form (LNF). LNF is a superclass of CNF that also allows rules whose right-hand sides have no consecutive nonterminals. The intuition is that the terminals provide anchors that limit the applicability of a given rule. For instance, consider the rule NP → the NPB of NNP. See Figure 2. Because the terminals constrain our choices, there are only two different application contexts. The implicit assumption is that input sentences will not repeat the same word more than a small constant number of times. If we make the explicit assumption that all words of an input sentence are unique, then there are $O(n^2)$ application contexts for a “no consecutive nonterminals” rule. Thus under this assumption, the running time of chart parsing is still $O(n^3)$ when applied to LNF grammars.

But once we make this assumption explicit, it becomes clear that we can go even further than LNF and still maintain the cubic bound on the runtime. Consider the rule NP → the JJ NPB of NNP. This rule is not LNF, but there are still only $O(n^2)$ application contexts, due to the anchoring effect of the terminals. In general, for a rule of the form X → γ, there are at most $O(n^p)$ application contexts, where p is the number of consecutive nonterminal pairs in the string X $\cdot\gamma\cdot$ X (where X is an arbitrary nonterminal). We refer to p as the *scope* of a rule. Thus chart parsing runs in time $O(n^{\mathsf{scope}(G)})$, where $\mathsf{scope}(G)$ is the maximum scope of any of the rules in CFG G. Specifically, any scope-3 grammar can be decoded in cubic time.

Like (DeNero et al., 2009), the target of our interest is synchronous context-free grammar (SCFG) decoding with rules extracted using the GHKM algorithm (Galley et al., 2004). In practice, it turns out that only a small percentage of the lexical rules in our system have scope greater than 3. By simply removing these rules from the grammar, we can maintain the cubic running time of chart parsing without any kind of binarization. This has three advantages. First, we do not inflate the grammar constant. Second, unlike (DeNero et al., 2009), we maintain the *synchronous* property of the grammar, and thus can integrate language model scoring into chart parsing. Finally, a system without binarized rules is considerably simpler to build and maintain. We show that this approach gives us better practical performance than a mature system that binarizes using the technique of (Zhang et al., 2006).

2 Preliminaries

Assume we have a global vocabulary of *symbols*, containing the reserved *substitution symbol* $\Diamond$. Define a *sentence* as a sequence of symbols. We will typically use space-delimited quotations to represent example sentences, e.g. “the fast jet ski” rather than $\langle$the, fast, jet, ski$\rangle$. We will use the dot operator to represent the concatenation of sentences, e.g. “the fast” · “jet ski” = “the fast jet ski”.

Define the *rank* of a sentence as the count of its $\Diamond$ symbols. We will use the notation $\textsc{sub}(s, s_1, ..., s_k)$ to denote the substitution of k sentences $s_1, ..., s_k$ into a k-rank sentence s. For instance, if s = “the $\Diamond$ $\Diamond$ of $\Diamond$”, then $\textsc{sub}(s, \text{“fast”}, \text{“jet ski”}, \text{“mr smith”})$ = “the fast jet ski of mr smith”.

To refer to a subsentence, define a *span* as a pair $[a, b]$ of nonnegative integers such that $a < b$. For a sentence $s = \langle s_1, s_2, ..., s_n\rangle$ and a span $[a, b]$ such that $b \leq n$, define $s_{[a,b]} = \langle s_{a+1}, ..., s_b\rangle$.

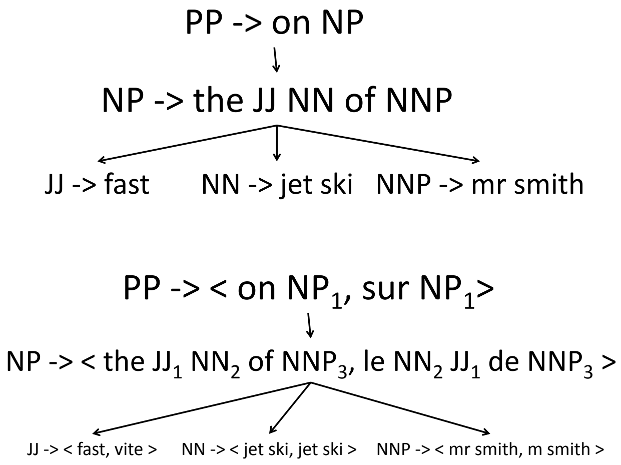

PP -> on NP
NP -> the JJ NN of NNP
JJ -> fast
NN -> jet ski
NNP -> mr smith
PP -> < on NP$_1$, sur NP$_1$>
NP -> < the JJ$_1$ NN$_2$ of NNP$_3$, le NN$_2$ JJ$_1$ de NNP$_3$ >
JJ -> < fast, vite >
NN -> < jet ski, jet ski >
NNP -> < mr smith, m smith >

Figure 3: An example CFG derivation (above) and an example SCFG derivation (below). Both derive the sentence SUB("on ◊", SUB("the ◊ ◊ of ◊", "fast", "jet ski", "mr smith")) = "on the fast jet ski of mr smith". The SCFG derivation simultaneously derives the auxiliary sentence "sur le jet ski vite de m smith".

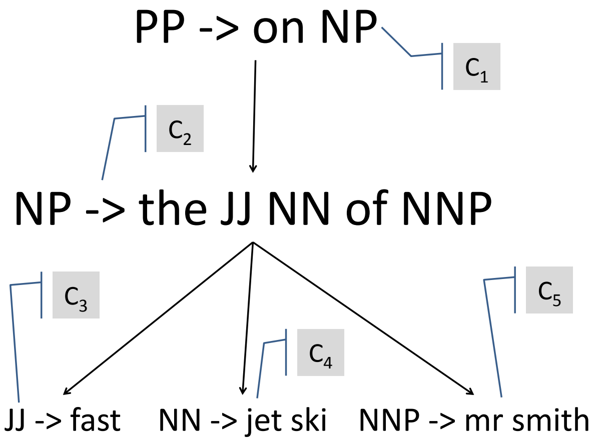


Figure 4: The cost of the CFG derivation in Figure 3 is $C_1 + C_2 + C_3 + C_4 + C_5$, where C_1 is the base cost of rule "PP → on NP", etc. Notice that this cost can be distributed locally to the nodes of the derivation.

3 Minimum Derivation Cost

Chart parsing solves a problem which we will refer to as Minimum Derivation Cost. Because we want our results to be applicable to both CFG decoding and SCFG decoding with an integrated language model, we will provide a somewhat more abstract formulation of chart parsing than usual.

In Figure 3, we show an example of a CFG derivation. A derivation is a tree of CFG rules, constructed so that the *preconditions* (the RHS nonterminals) of any rule match the *postconditions* (the LHS nonterminal) of its child rules. The purpose of a derivation is to *derive* a sentence, which is obtained through recursive substitution. In the example, we substitute "fast", "jet ski", and "mr smith" into the lexical *pattern* "the ◊ ◊ of ◊" to obtain "the fast jet ski of mr smith". Then we substitute this result into the lexical pattern "on ◊" to obtain "on the fast jet ski of mr smith".

The cost of a derivation is simply the sum of the base costs of its rules. Thus the cost of the CFG derivation in Figure 3 is $C_1 + C_2 + C_3 + C_4 + C_5$, where C_1 is the base cost of rule "PP → on NP", etc. Notice that this cost can be distributed locally to the nodes of the derivation (Figure 4).

An SCFG derivation is similar to a CFG derivation, except that it simultaneously derives two sentences. For instance, the SCFG derivation in Figure 3 derives the sentence pair ⟨ "on the fast jet ski of mr smith", "sur le jet ski vite de m smith" ⟩. In machine translation, often we want the cost of the SCFG derivation to include a language model cost for this second sentence. For example, the cost of the SCFG derivation in Figure 3 might be $C_1+C_2+C_3+C_4+C_5+LM(\text{sur le})+LM(\text{le jet})+LM(\text{jet ski})+LM(\text{ski de}) + LM(\text{de m}) + LM(\text{m smith})$, where LM is the negative log of a 2-gram language model. This new cost function can also be distributed locally to the nodes of the derivation, as shown in Figure 5. However, in order to perform the local computations, we need to pass information (in this case, the LM boundary words) up the tree. We refer to this extra information as *carries*. Formally, define a *carry* as a sentence of rank 0.

In order to provide a chart parsing formulation that applies to both CFG decoding and SCFG decoding with an integrated language model, we need abstract definitions of rule and derivation that capture the above concepts of *pattern*, *postcondition*, *preconditions*, *cost*, and *carries*.

3.1 Rules

Define a *rule* as a tuple $\langle k, s^*, X, \pi, \Gamma, c \rangle$, where k is a nonnegative integer called the *rank*, s^* is a rank-k

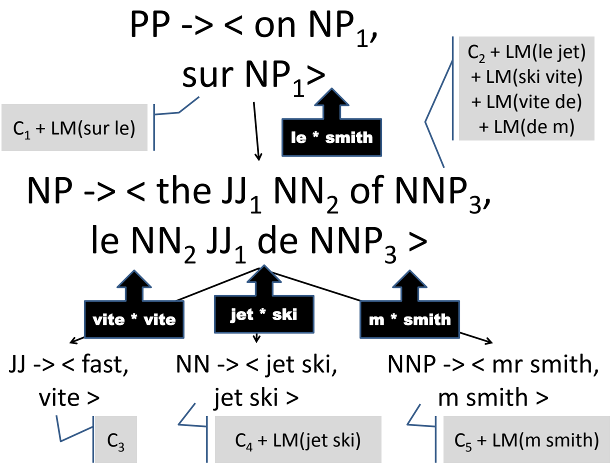

PP -> < on NP1, sur NP1>
C1 + LM(sur le)
le * smith
C2 + LM(le jet) + LM(ski vite) + LM(vite de) + LM(de m)
NP -> < the JJ1 NN2 of NNP3, le NN2 JJ1 de NNP3 >
vite * vite
jet * ski
m * smith
JJ -> < fast, vite >
NN -> < jet ski, jet ski >
NNP -> < mr smith, m smith >
C3
C4 + LM(jet ski)
C5 + LM(m smith)

Figure 5: The cost of the SCFG derivation in Figure 3 (with an integrated language model score) can also be distributed to the nodes of the derivation, but to perform the local computations, information must be passed up the tree. We refer to this extra information as a *carry*.

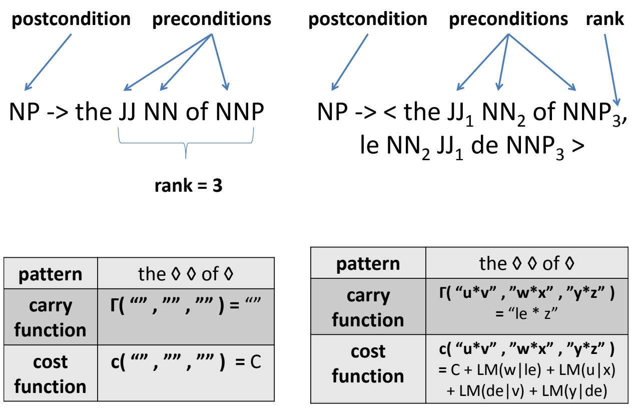


Figure 6: Deconstruction of a CFG rule (left) and SCFG rule (right) according to the definition of rule in Section 3.1. The carry function of the SCFG rule computes boundary words for a 2-gram language model. In the cost functions, C is a real number and LM returns the negative log of a language model query.

sentence called the *pattern* [1], X is a symbol called the *postcondition*, π is a k-length sentence called the *preconditions*, Γ is a function (called the *carry function*) that maps a k-length list of carries to a carry, and c is a function (called the *cost function*) that maps a k-length list of carries to a real number. Figure 6 shows a CFG and an SCFG rule, deconstructed according to this definition. [2] Note that the CFG rule has trivial cost and carry functions that map everything to a constant. We refer to such rules as *simple*.

We will use $\mathsf{post}(r)$ to refer to the postcondition of rule r, and $\mathsf{pre}(r, i)$ to refer to the i^{th} precondition of rule r.

Finally, define a *grammar* as a finite set of rules. A grammar is *simple* if all its rules are simple.

3.2 Derivations

For a grammar R, define $\mathsf{deriv}(R)$ as the smallest set that contains every tuple $\langle r, \delta_1, ..., \delta_k \rangle$ satisfying the following conditions:

- $r \in R$ is a k-rank rule
- $\delta_i \in \mathsf{deriv}(R)$ for all $1 \leq i \leq k$
- $\mathsf{pre}(r, i) = \mathsf{post}(r_i)$ for all $1 \leq i \leq k$, where r_i is the first element of tuple δ_i.

An *R–derivation* is an element of $\mathsf{deriv}(R)$. Consider a derivation $\delta = \langle r, \delta_1, ..., \delta_k \rangle$, where rule $r = \langle k, s^*, X, \pi, \Gamma, c \rangle$. Define the following properties:

$$\mathsf{post}(\delta) = \mathsf{post}(r)$$
$$\mathsf{sent}(\delta) = \textsc{sub}(s^*, \mathsf{sent}(\delta_1), ..., \mathsf{sent}(\delta_k))$$
$$\mathsf{carry}(\delta) = \Gamma(\mathsf{carry}(\delta_1), ..., \mathsf{carry}(\delta_k))$$
$$\mathsf{cost}(\delta) = c(\mathsf{carry}(\delta_1), ..., \mathsf{carry}(\delta_k)) + \sum_{j=1}^{k} \mathsf{cost}(\delta_j)$$

In words, we say that derivation δ *derives* sentence $\mathsf{sent}(\delta)$. If for some span σ of a particular sentence s, it holds that $\mathsf{sent}(\delta) = s_\sigma$, then we will say that δ is a derivation *over* span σ.

3.3 Problem Statement

The Minimum Derivation Cost problem is the following. Given a set R of rules and an input sentence

[1] For simplicity, we also impose the condition that “◊” is not a valid pattern. This is tantamount to disallowing unary rules.

[2] One possible point of confusion is why the pattern of the SCFG rule refers only to the primary sentence, and not the auxiliary sentence. To reconstruct the auxiliary sentence from an SCFG derivation in practice, one would need to augment the abstract definition of rule with an auxiliary pattern. However this is not required for our theoretical results.

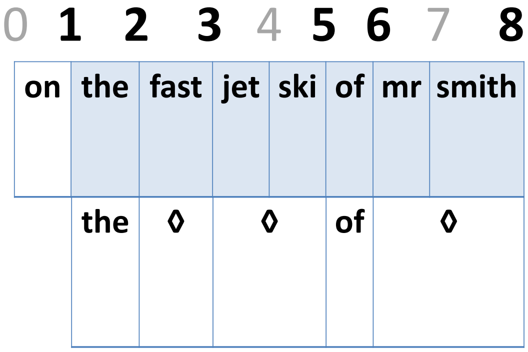


Figure 7: An application context for the pattern "the ◊ ◊ of ◊" and the sentence "on the fast jet ski of mr smith".

s, find the minimum cost of any R–derivation that derives s. In other words, compute:

$$\mathsf{MinDCost}(R, s) \triangleq \min_{\delta \in \mathsf{deriv}(R) | \mathsf{sent}(\delta) = s} \mathsf{cost}(\delta)$$

4 Application Contexts

Chart parsing solves Minimum Derivation Cost via dynamic programming. It works by building derivations over increasingly larger spans of the input sentence s. Consider just one of these spans σ. How do we build a derivation over that span?

Recall that a derivation takes the form $\langle r, \delta_1, ..., \delta_k \rangle$. Given the rule r and its pattern s^*, we need to choose the subderivations δ_i such that $\text{SUB}(s^*, \mathsf{sent}(\delta_1), ..., \mathsf{sent}(\delta_k)) = s_\sigma$. To do so, we must match the pattern to the span, so that we know which subspans we need to build the subderivations over. Figure 7 shows a matching of the pattern "the ◊ ◊ of ◊" to span $[1, 8]$ of the sentence "on the fast jet ski of mr smith". It tells us that we can build a derivation over span $[1, 8]$ by choosing this rule and subderivations over subspans $[2, 3]$, $[3, 5]$, and $[6, 8]$.

We refer to these matchings as *application contexts*. Formally, given two sentences s^* and s of respective lengths m and n, define an $\langle s^*, s \rangle$–*context* as an monotonically increasing sequence $\langle x_0, x_1, ..., x_m \rangle$ of integers between 0 and n such that for all i:

$$s^*_{[i-1,i]} \neq \lozenge \text{ implies that } s^*_{[i-1,i]} = s_{[x_{i-1}, x_i]}$$

The context shown in Figure 7 is $\langle 1, 2, 3, 5, 6, 8 \rangle$. Use $\mathsf{cxt}(s^*, s)$ to denote the set of all $\langle s^*, s \rangle$–contexts.

An $\langle s^*, s \rangle$–context $x = \langle x_0, x_1, ..., x_m \rangle$ has the following properties:

$$\mathsf{span}(x; s^*, s) = [x_0, x_m]$$
$$\mathsf{subspans}(x; s^*, s) = \langle [x_0, x_1], ..., [x_{m-1}, x_m] \rangle$$

Moreover, define $\mathsf{varspans}(x; s^*, s)$ as the subsequence of $\mathsf{subspans}(x; s^*, s)$ including only $[x_{i-1}, x_i]$ such that $s^*_{[i-1,i]} = \lozenge$. For the context x shown in Figure 7:

$$\mathsf{span}(x; s^*, s) = [1, 8]$$
$$\mathsf{subspans}(x; s^*, s) = \langle [1, 2], [2, 3], [3, 5], [5, 6], [6, 8] \rangle$$
$$\mathsf{varspans}(x; s^*, s) = \langle [2, 3], [3, 5], [6, 8] \rangle$$

An application context $x \in \mathsf{cxt}(s^*, s)$ tells us that we can build a derivation over $\mathsf{span}(x)$ by choosing a rule with pattern s^* and subderivations over each span in $\mathsf{varspans}(x; s^*, s)$.

5 Chart Parsing Algorithm

We are now ready to describe the chart parsing algorithm. Consider a span σ of our input sentence s and assume that we have computed and stored all derivations over any subspan of σ. A naive way to compute the minimum cost derivation over span σ is to consider every possible derivation:

1. Choose a rule $r = \langle k, s^*, X, \pi, \Gamma, c \rangle$.
2. Choose an application context $x \in \mathsf{cxt}(s^*, s)$ such that $\mathsf{span}(x; s^*, s) = \sigma$.
3. For each subspan $\sigma_i \in \mathsf{varspans}(x; s^*, s)$, choose a subderivation δ_i such that $\mathsf{post}(\delta_i) = \mathsf{pre}(r, i)$.

The key observation here is the following. In order to score such a derivation, we did not actually need to know each subderivation in its entirety. We merely needed to know the following information about it: (a) the subspan that it derives, (b) its postcondition, (c) its carry.

Chart parsing takes advantage of the above observation to avoid building all possible derivations. Instead it groups together derivations that share a common subspan, postcondition, and carry, and records only the minimum cost for each equivalence class. It records this cost in an associative map referred to as the *chart*.

Specifically, assume that we have computed and stored the minimum cost of every derivation class $\langle \sigma', X', \gamma' \rangle$, where X' is a postcondition, γ' is a carry, and σ' is a proper subspan of σ. Chart parsing computes the minimum cost of every derivation class $\langle \sigma, X, \gamma \rangle$ by adapting the above naive method as follows:

1. Choose a rule $r = \langle k, s^*, X, \pi, \Gamma, c \rangle$.

2. Choose an application context $x \in \mathsf{cxt}(s^*, s)$ such that $\mathsf{span}(x; s^*, s) = \sigma$.

3. For each subspan $\sigma_i \in \mathsf{varspans}(x; s^*, s)$, **choose a derivation class** $\langle \sigma_i, X_i, \gamma_i \rangle$ **from the chart** such that $X_i = \mathsf{pre}(r, i)$.

4. **Update**[3] **the cost of derivation class** $\langle \sigma, \mathsf{post}(r), \Gamma(\gamma_1, ..., \gamma_k) \rangle$ **with:**

$$c(\gamma_1, ..., \gamma_k) + \sum_{i=1}^{k} \mathsf{chart}[\sigma_i, X_i, \gamma_i]$$

where $\mathsf{chart}[\sigma_i, X_i, \gamma_i]$ **refers to the stored cost of derivation class** $\langle \sigma_i, X_i, \gamma_i \rangle$.

By iteratively applying the above method to all subspans of size 1, 2, etc., chart parsing provides an efficient solution for the Minimum Derivation Cost problem.

6 Runtime Analysis

At the heart of chart parsing is a single operation: the updating of a value in the chart. The running time is linear in the number of these chart updates. [4] The typical analysis counts the number of chart updates *per span*. Here we provide an alternative analysis that counts the number of chart updates *per rule*. This provides us with a finer bound with practical implications.

Let r be a rule with rank k and pattern s^*. Consider the chart updates involving rule r. There is (potentially) an update for every choice of (a) span, (b) application context, and (c) list of k derivation classes. If we let $\mathfrak{C}$ be the set of possible carries, then this means there are at most $|\mathsf{cxt}(s^*, s)| \cdot |\mathfrak{C}|^k$ updates involving rule r. [5] If we are doing beam decoding (i.e. after processing a span, the chart keeps only the B items of lowest cost), then there are at most $|\mathsf{cxt}(s^*, s)| \cdot B^k$ updates.

We can simplify the above by providing an upper bound for $|\mathsf{cxt}(s^*, s)|$. Define an *ambiguity* as the sentence "$\Diamond$ $\Diamond$", and define $\mathsf{scope}(s^*)$ as the number of ambiguities in the sentence "$\Diamond$" $\cdot s^* \cdot$ "$\Diamond$". The following bound holds:

Lemma 1. *Assume that a zero-rank sentence s does not contain the same symbol more than once. Then* $|\mathsf{cxt}(s^*, s)| \leq |s|^{\mathsf{scope}(s^*)}$.

Proof. Suppose s^* and s have respective lengths m and n. Consider $\langle x_0, x_1, ..., x_m \rangle \in \mathsf{cxt}(s^*, s)$. Let I be the set of integers i between 1 and m such that $s^*_i \neq \Diamond$ and let I^+ be the set of integers i between 0 and $m-1$ such that $s^*_{i+1} \neq \Diamond$. If $i \in I$, then we know the value of x_i, namely it is the unique integer j such that $s_j = s^*_i$. Similarly, if $i \in I^+$, then the value of x_i must be the unique integer j such that $s_j = s^*_{i+1}$. Thus the only nondetermined elements of context x_i are those for which $i \notin I \cup I^+$. Hence $|\mathsf{cxt}(s^*, s)| \leq |s|^{\{0,1,...,m\} - I - I^+} = |s|^{\mathsf{scope}(s^*)}$. □

Hence, under the assumption that the input sentence s does not contain the same symbol more than once, then there are at most $|s|^{\mathsf{scope}(s^*)} \cdot |\mathfrak{C}|^k$ chart updates involving a rule with pattern s^*.

For a rule r with pattern s^*, define $\mathsf{scope}(r) = \mathsf{scope}(s^*)$. For a grammar R, define $\mathsf{scope}(R) = \max_{r \in R} \mathsf{scope}(r)$ and $\mathsf{rank}(R) = \max_{r \in R} \mathsf{rank}(r)$.

Given a grammar R and an input sentence s, the above lemma tells us that chart parsing makes

[3] Here, *update* means "replace the cost associated with the class if the new cost is lower."

[4] This assumes that you can linearly enumerate the relevant updates. One convenient way to do this is to frame the enumeration problem as a search space, e.g. (Hopkins and Langmead, 2009)

[5] For instance, in SCFG decoding with an integrated j-gram language model, a carry consists of $2(j-1)$ boundary words. Generally it is assumed that there are $O(n)$ possible choices for a boundary word, and hence $O(n^{2(j-1)})$ possible carries.

$O(|s|^{\mathsf{scope}(R)} \cdot |\mathfrak{C}|^{\mathsf{rank}(R)})$ chart updates. If we restrict ourselves to beam search, than chart parsing makes $O(|s|^{\mathsf{scope}(R)})$ chart updates. [6]

6.1 On the Uniqueness Assumption

In practice, it will not be true that each input sentence contains only unique symbols, but it is not too far removed from the practical reality of many use cases, for which relatively few symbols repeat themselves in a given sentence. The above lemma can also be relaxed to assume only that there is a constant upper bound on the multiplicity of a symbol in the input sentence. This does not affect the O-bound on the number of chart updates, as long as we further assume a constant limit on the length of rule patterns.

7 Scope Reduction

From this point of view, CNF binarization can be viewed as a specific example of *scope reduction*. Suppose we have a grammar R of scope p. See Figure 8. If we can find a grammar $\hat{R}$ of scope $\hat{p} < p$ which is "similar" to grammar R, then we can decode in $O(n^{\hat{p}})$ rather than $O(n^p)$ time.

We can frame the problem by assuming the following parameters:

- a grammar R
- a desired scope p
- a loss function Λ that returns a (non-negative real-valued) score for any two grammars R and $\hat{R}$; if $\Lambda(R, \hat{R}) = 0$, then the grammars are considered to be equivalent

A *scope reduction method with loss* λ finds a grammar $\hat{R}$ such that $\mathsf{scope}(\hat{R}) \leq p$ and $\Lambda(R, \hat{R}) = \lambda$. A scope reduction method is *lossless* when its loss is 0.

In the following sections, we will use the loss function:

$$\Lambda(R, \hat{R}) = |\mathsf{MinDCost}(R, s) - \mathsf{MinDCost}(\hat{R}, s)|$$

where s is a fixed input sentence. Observe that if $\Lambda(R, \hat{R}) = 0$, then the solution to the Minimum Derivation Cost problem is the same for both R and $\hat{R}$. [7]

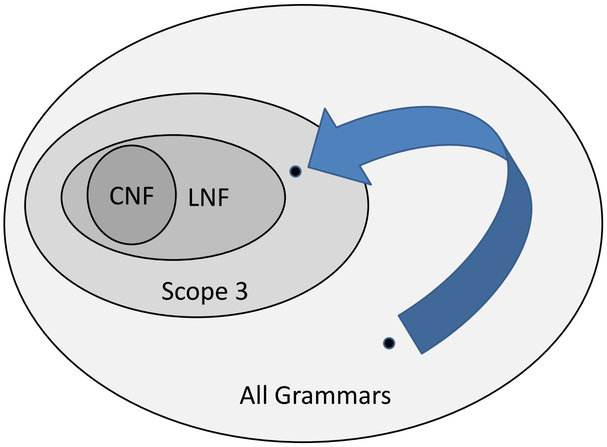


Figure 8: The "scope reduction" problem. Given a grammar of large scope, find a similar grammar of reduced scope.

7.1 CNF Binarization

A rule r is *CNF* if its pattern is "$\Diamond\ \Diamond$" or "x", where x is any non-substitution symbol. A grammar is *CNF* if all of its rules are CNF. Note that the maximum scope of a CNF grammar is 3.

CNF binarization is a deterministic process that maps a simple grammar to a CNF grammar. Since binarization takes subcubic time, we can decode with any grammar R in $O(n^3)$ time by converting R to CNF grammar $\hat{R}$, and then decoding with $\hat{R}$. This is a lossless scope reduction method.

What if grammar R is not simple? For SCFG grammars, (Zhang et al., 2006) provide a scope reduction method called *synchronous binarization* with quantifiable loss. Synchronous binarization selects a "binarizable" subgrammar R' of grammar R, and then converts R' into a CNF grammar $\hat{R}$. The cost and carry functions of these new rules are constructed such that the conversion from R' to $\hat{R}$ is a lossless scope reduction. Thus the total loss of the method is $|\mathsf{MinDCost}(R, s) - \mathsf{MinDCost}(R', s)|$. Fortunately, they find in practice that R' usually contains the great majority of the rules of R, thus they

[6] Assuming $\mathsf{rank}(R)$ is bounded by a constant.

[7] Note that if we want the actual derivation and not just its cost, then we need to specify a more finely grained loss function. This is omitted for clarity and left as an exercise.

a ◊ ◊
◊ ◊ a
a ◊ ◊ b
◊ a ◊ ◊
◊ ◊ a ◊
◊ ◊ ◊ a
a ◊ ◊ ◊
a b ◊ ◊
◊ ◊ a b
a ◊ ◊ b ◊
a ◊ ◊ b c

a b ◊ ◊ c
◊ a ◊ ◊ b
a ◊ b ◊ ◊
◊ ◊ a ◊ b
◊ ◊ ◊ a b
a ◊ ◊ ◊ b
a ◊ ◊ b ◊ c
a ◊ ◊ ◊ ◊ b
a ◊ ◊ b c ◊ ◊ d
a ◊ ◊ ◊ ◊ b ◊ c ◊ d
a ◊ ◊ b ◊ ◊ c ◊ ◊ d

Figure 9: A selection of rule patterns that are scope ≤ 3 but not LNF or CNF.

assert that this loss is negligable.

A drawback of their technique is that the resulting CNF grammar contains many more rules and postconditions than the original grammar. These constant factors do not impact asymptotic performance, but do impact practical performance.

7.2 Lexical Normal Form

Concerned about this inflation of the grammar constant, (DeNero et al., 2009) consider a superset of CNF called Lexical Normal Form (LNF). A rule is *LNF* if its pattern does not contain an ambiguity as a proper subsentence (recall that an ambiguity was defined to be the sentence "◊ ◊"). Like CNF, the maximum scope of an LNF grammar is 3. In the worst case, the pattern s^* is "◊ ◊", in which case there are three ambiguities in the sentence "◊" $\cdot s^* \cdot$ "◊".

(DeNero et al., 2009) provide a lossless scope reduction method that maps a simple grammar to an LNF grammar, thus enabling cubic-time decoding. Their principal objective is to provide a scope reduction method for SCFG that introduces fewer postconditions than (Zhang et al., 2006). However unlike (Zhang et al., 2006), their method *only* addresses simple grammars. Thus they cannot integrate LM scoring into their decoding, requiring them to rescore the decoder output with a variant of cube growing (Huang and Chiang, 2007).

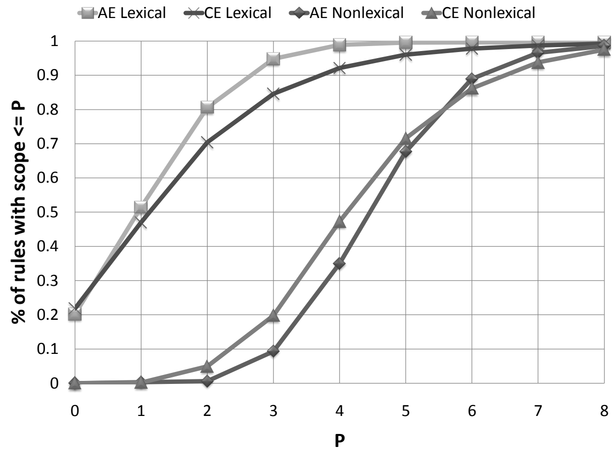


Figure 10: Breakdown of rules by scope (average per sentence in our test sets). In practice, most of the lexical rules applicable to a given sentence (95% for Arabic-English and 85% for Chinese-English) are scope 3 or less.

7.3 Scope Pruning

To exercise the power of the ideas presented in this paper, we experimented with a third (and very easy) scope reduction method called *scope pruning*. If we consider the entire space of scope-3 grammars, we see that it contains a much richer set of rules than those permitted by CNF or LNF. See Figure 9 for examples. Scope pruning is a lossy scope reduction method that simply takes an arbitrary grammar and prunes all rules with scope greater than 3. By not modifying any rules, we preserve their cost and carry functions (enabling integrated LM decoding), without increasing the grammar constant. The practical question is: how many rules are we typically pruning from the original grammar?

We experimented with two pretrained syntax-based machine translation systems with rules extracted via the GHKM algorithm (Galley et al., 2004). The first was an Arabic-English system, with rules extracted from 200 million words of parallel data from the NIST 2008 data collection, and with a 4-gram language model trained on 1 billion words of monolingual English data from the LDC Gigaword corpus. We evaluated this system's performance on the NIST 2008 test corpus, which consists of 1357 Arabic sentences from a mixture of newswire and web domains, with four English reference translations. The second system was a Chinese-

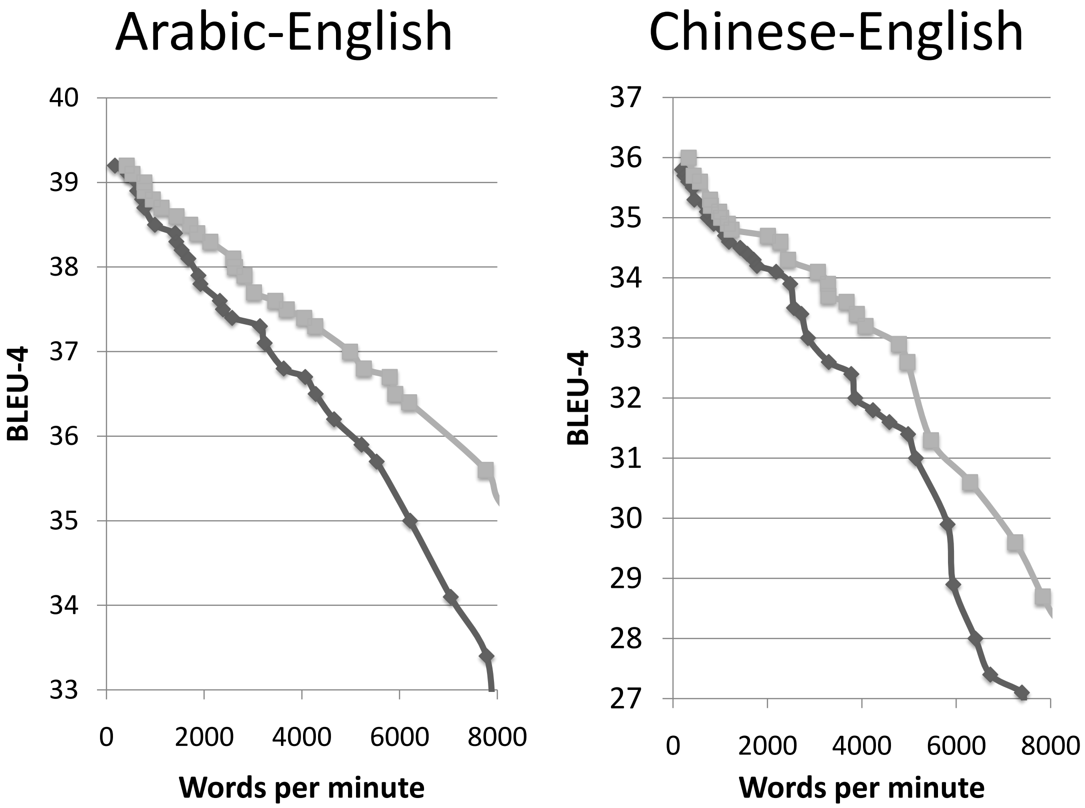


Figure 11: Speed-quality tradeoff curves comparing the baseline scope reduction method of synchronous binarization (dark gray diamonds) with scope-3 pruning (light gray squares).

English system, with rules extracted from 16 million words of parallel data from the mainland-news domain of the LDC corpora, and with a 4-gram language model trained on monolingual English data from the AFP and Xinhua portions of the LDC Gigaword corpus. We evaluated this system's performance on the NIST 2003 test corpus, which consists of 919 Chinese sentences, with four English reference translations. For both systems, we report BLEU scores (Papineni et al., 2002) on untokenized, recapitalized output.

In practice, how many rules have scope greater than 3? To answer this question, it is useful to distinguish between *lexical* rules (i.e. rules whose patterns contain at least one non-substitution symbol) and *non-lexical* rules. Only a subset of lexical rules are potentially applicable to a given input sentence. Figure 10 shows the scope profile of these applicable rules (averaged over all sentences in our test sets). Most of the lexical rules applicable to a given sentence (95% for Arabic-English, 85% for Chinese-English) are scope 3 or less. [8] Note, however, that scope pruning also prunes a large percentage of non-lexical rules.

Figure 11 compares scope pruning with the baseline technique of synchronous binarization. To generate these speed-quality tradeoff curves, we decoded the test sets with 380 different beam settings. We then plotted the hull of these 380 points, by eliminating any points that were dominated by another (i.e. had better speed and quality). We found that this simple approach to scope reduction produced a better speed-quality tradeoff than the much more complex synchronous binarization. [9]

[8] For contrast, the corresponding numbers for LNF are 64% and 53%, respectively.

[9] We also tried a hybrid approach in which we scope-pruned

8 Conclusion

In this paper, we made the following contributions:

- We provided an abstract formulation of chart parsing that generalizes CFG decoding and SCFG decoding with an integrated LM.
- We framed *scope reduction* as a first-class abstract problem, and showed that CNF binarization and LNF binarization are two specific solutions to this problem, each with their respective advantages and disadvantages.
- We proposed a third scope reduction technique called *scope pruning*, and we showed that it can outperform synchronous CNF binarization for particular use cases.

Moreover, this work gives formal expression to the extraction heuristics of hierarchical phrase-based translation (Chiang, 2007), whose directive not to extract SCFG rules with adjacent nonterminals can be viewed as a preemptive pruning of rules with scope greater than 2 (more specifically, the pruning of non-LNF lexical rules). In general, this work provides a framework in which different approaches to *tractability-focused* grammar construction can be compared and discussed.

References

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

John DeNero, Mohit Bansal, Adam Pauls, and Dan Klein. 2009. Efficient parsing for transducer grammars. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT/NAACL.*

Mark Hopkins and Greg Langmead. 2009. Cube pruning as heuristic search. In *Proceedings of EMNLP*.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of ACL.*

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 Association for Computational Linguistics*, pages 311–318.

Daniel Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 256–263.

the lexical rules and synchronously binarized the non-lexical rules. This had a similar performance to scope-pruning all rules. The opposite approach of scope-pruning the lexical rules and synchronously binarizing the non-lexical rules had a similar performance to synchronous binarization.

Example-based Paraphrasing for Improved Phrase-Based Statistical Machine Translation

Aurélien Max
LIMSI-CNRS & Univ. Paris Sud
Orsay, France
aurelien.max@limsi.fr

Abstract

In this article, an original view on how to improve phrase translation estimates is proposed. This proposal is grounded on two main ideas: first, that appropriate examples of a given phrase should participate more in building its translation distribution; second, that paraphrases can be used to better estimate this distribution. Initial experiments provide evidence of the potential of our approach and its implementation for effectively improving translation performance.

1 Introduction

Phrase translation estimation in Statistical Phrase-based Translation (Koehn et al., 2003) is hampered by the availability of both too many and too few training instances. Recent results on tera-scale SMT (Lopez, 2008) show that access to many training examples[1] can lead to significant improvements in translation quality. Also, providing indirect training instances via synonyms or paraphrases for previously unseen phrases can result in gains in translation quality, which are more apparent when little training data is originally available (Callison-Burch et al., 2006; Marton et al., 2009; Mirkin et al., 2009; Aziz et al., 2010). Although there is a consensus on the importance of using more parallel data in SMT, it has never been formally shown that all additional training instances are actually useful in predicting contextually appropriate translation hypotheses.

Attempts at limiting training parallel sentences to those resembling test data through thematic adaptation (Hildebrand et al., 2005) indeed confirm that large quantities of training data cannot compensate for the requirement for contextually appropriate training instances. In fact, it is important that phrase translation models adequatly reflect contextual preferences for each phrase occurrence in a text. A variety of recent works have used dynamically adapted translation models, where each phrase occurrence has its own translation distribution (Carpuat and Wu, 2007; Stroppa et al., 2007; Max et al., 2008; Gimpel and Smith, 2008; Haque et al., 2009) derived from local contextual information in the training examples.[2] These approaches are supported by the study of (Wisniewski et al., 2010) which shows that phrase-based SMT systems are expressive enough to achieve very high translation performance and therefore suggests a better scoring of phrases.

The apparent tradeoff between the number of training examples and their appropriateness in each indivual context naturally asks for means of increasing the number of appropriate examples. Exploiting comparable corpora for acquiring translation equivalents (Munteanu and Marcu, 2005; Abdul-Rauf and Schwenk, 2009) offers interesting prospects to this issue, but so far focus has not been so much on context appropriateness as on globally increasing the number of biphrase examples.

[1] To be more accurate, works such as that of (Lopez, 2008) have recourse to random sampling to build models of a manageable size in a reasonable amount of time.

[2] The study of (Carpuat, 2009) shows that the one translation per discourse hypothesis holds in some cases, but to our knowledge no SMT systems have attempted to exploit it yet. However, in our view, this finding does not contradict the need for estimating translation distributions at the individual phrase level, but they should be integrated as additional information.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 656–666,
MIT, Massachusetts, USA, 9-11 October 2010.

The approach we take in this article is motivated by the fact that natural language allows for multiple text views on a given content, and that if two phrases are good paraphrases in context, then considering appropriate training examples of one of the phrases could provide larger quantities of training data for translating the other. In other words, we hypothesize that there may be more training data to learn a phrase's translations in a bilingual corpus than what SMT approaches typically use.

In contrast to previous attempts at using paraphrases to improve Statistical Machine Translation, which require external data in the form of additional parallel bilingual corpora (Callison-Burch et al., 2006), monolingual corpora (Marton et al., 2009), lexico-semantic resources (Mirkin et al., 2009; Aziz et al., 2010), or sub-sentential (Resnik et al., 2010) or sentential paraphrases of the input (Schroeder et al., 2009), the approach we take here can be endogenous with respect to the original training data. It also significantly departs from previous work in that paraphrasing is not simply considered as a way of finding alternative wordings that can be translated given the original training data for out-of-vocabulary phrases only (Callison-Burch et al., 2006; Marton et al., 2009; Mirkin et al., 2009; Aziz et al., 2010), but as a means to better estimate translations for *any* possible phrase. Also, as opposed to the work by (Schroeder et al., 2009; Onishi et al., 2010; Du et al., 2010), we do not encode paraphrases into input lattices to have them compete against each other to belong to the source sentential paraphrase that will lead to the highest scoring output sentence[3]. Instead, we make use of all contextually appropriate paraphrases of a source phrase, which *collectively evaluate the quality of each translation for that phrase*.

This work can thus be seen as a contribution towards shifting from global phrase translation distributions to contextual translation distributions for contextually equivalent source units. The remainder of this paper is organized as followed. In section 2 we review relevant previous works and discuss how they differ from our approach. Section 3 provides a description of the details of our approach. We describe an experimental setup in section 4 and comment on our results. Finally, we discuss our future work in section 5.

2 Relation to previous work

2.1 Contextual estimation of phrase translations

In standard approaches to phrase-based SMT, evidence of a translation is accumulated uniformely every time it is found associated with a source phrase in the training corpus. In addition to the fact that errors in automatic word alignment and non literal translations often produce useless biphrases, this results in rare but appropriate translations being very unlikely to be considered during decoding. Some approaches on source context modelling (Carpuat and Wu, 2007; Stroppa et al., 2007; Max et al., 2008; Haque et al., 2009) build classifiers offline for the phrases in a test set, so that context similarity can for example reinforce scores associated with rare but appropriate translations. However, heavy offline computation makes scaling to larger corpora an issue. Other approaches (Callison-Burch et al., 2005; Lopez, 2008) instead focus on accessing very large corpora. Indexing by suffix arrays is used to allow fast access to phrase instances in the corpus, and random sampling to avoid collecting the full set of examples has been shown to perform well. However, these approaches consider all instances of a phrase as equivalent for the estimation of its translations.

These works converge on the need for accessing a sufficient number of examples that are relevant for any source phrase in context, fast enough to permit on-the-fly phrase table building. This paper proposes an intermediate step: the full set of phrase examples is found efficiently, and a measure of the adequacy of each example with a phrase in context provides evidence for its translation that depends on this value of adequacy. In this way, the translation associated with an example for a different sense of a polysemous word would in the best scenario only be considered marginally when computing the translation distribution. As in most previous works, adequacy can be approximated by context similarity between phrase occurrences and training examples.

Ideally, one would stop extracting examples when enough appropriate examples have been found to estimate a reliable translation distribution. (Callison-

[3] This highly depends on how well estimated translations for each independent paraphrase are.

sample size	100	500	1K	5K	10k	50k	unlim.
BLEU score	28.8	28.8	28.8	28.9	29.1	28.9	29.0

Figure 1: Effect of number of samples on translation quality (measured on German to English translation on Europarl data) reported by (Callison-Burch et al., 2005)

Burch et al., 2005) measured the impact on translation quality of the sample size in random sampling of source phrase examples in the training corpus to estimate a phrase's translation probabilities. As Table 1 shows, quality (in terms of BLEU scores) almost remains constant for samples of size 100 or more. This apparent confirmation of the efficiency of random sampling is backed up by the authors with the following possible explanations: 1) the most probable translations remain the same for different sample sizes; 2) misestimated probabilities are ruled out by the target language model; and 3) longer or less frequent phrases, which are not affected by sampling, are preferred. However, as said previously, random sampling cannot guarantee that contextually-appropriate examples are selected. In fact, (Lopez, 2008) points out to using discriminatively trained models with contextual features of source phrases in conjunction with phrase sampling as an open problem. This work does not attempt to directly address it, but instead resorts to complete analysis of the training data to guarantee that all contextually-appropriate examples are considered.

2.2 Using paraphrases for translating

For some phrases, not enough examples can be found in the training corpus to estimate reliable translation probabilities in context. In such cases, one might be interested in finding more appropriate examples, which seems at first impossible using the sole original bilingual corpus. We can in fact consider the set of source phrases that have similar translations in context. This set is roughly made up of a subset of what can be referred to as *paraphrases*. One possible approach to extract local (i.e. phrasal) paraphrases precisely exploits similarity *on the target side* in another language by extracting source phrases that share common translations (Bannard and Callison-Burch, 2005), but recent approaches have combined this approach with similarity computation in the "source" (i.e. original) language (Callison-Burch, 2008; Max, 2008; Kok and Brockett, 2010). Figure 2 provides examples of English paraphrases obtained by automatically pivoting via French. As can be seen, some examples would be clearly useful to better estimate translations of the original source phrase: (*Balkan War* ↔ *war in the Balkans*) are syntactic variants that can generally substitute with each other, (*Balkan War* ↔ *Balkans war*) are character-level variants[4]. Other examples, however, clearly illustrate the need for validation in context: (*Dalai Lama's* ↔ *of the Dalai Lama*) require different syntactic contexts, and (*I don't see* ↔ *I do not believe*) are only interchangeable in specific semantic contexts.

Source phrase	**Paraphrases**
Balkan War	*Balkan war* (0.25) *Balkans War* (0.125) *Balkans* (0.125) *Balkans war* (0.125) *war in the Balkans* (0.125) *Balkan conflict* (0.125)
British forces	*British troops* (0.29) *British armed_forces* (0.19)
Czech president	*President of the Czech Republic* (0.5)
Dalai Lama's	*of the Dalai Lama* (0.27)
I don't see	*I do not believe* (0.18) *I do not think* (0.18) *I do not see* (0.15)

Figure 2: Examples of paraphrases obtained by pivoting via French; values indicate *paraphrase probability* as defined in (Bannard and Callison-Burch, 2005).

Previous attempts at exploiting paraphrases in SMT have first concentrated on obtaining translations for phrases absent from the training corpus (Callison-Burch et al., 2006; Marton et al., 2009; Mirkin et al., 2009)[5], with modest gains in translation performance as measured by automatic metrics. (Callison-Burch et al., 2006) obtain paraphrases by pivoting via additional bilingual corpora and use the translations of known paraphrases to translate unseen phrases, which requires that the additional bilingual corpora contain the unseen source phrases and that some of the extracted paraphrases be present in the original corpus. (Marton et al.,

[4] To our knowledge, most implementations of SMT decoders do not integrate flexible matching of phrases.

[5] The work by (Mirkin et al., 2009) in fact considers both paraphrases and entailed texts to increase the number of properly translated texts.

2009) proceed similarly but obtain their paraphrases from comparatively much larger monolingual corpora by following the *distributionality hypothesis*. In both cases, gains are only obtained in very specific conditions where very few training data are available and where useful additional knowledge can be brought in from external resources. Furthermore, the described implementations do not consider acceptability of the paraphrases in context, as their underlying hypothesis is that it might be more desirable to translate some paraphrase than not to translate a given phrase.[6] In contrast, the work by (Mirkin et al., 2009) attempts to model context when using replacements for words (synonyms or hypernyms).

The natural next step that we take here is to exploit the complementarity of the original bilingual training corpus for finding paraphrases and the monolingual (source) side of the same corpus for validating them in context. Furthermore, our focus here is not on paraphrasing unseen phrases[7], but possibly any phrase, or any phrase seen less than a given number of times, or any types of difficult-to-translate phrases (Mohit and Hwa, 2007).

The recent work of (Resnik et al., 2010) uses crowdsourcing to obtain paraphrases for source phrases corresponding to mistranslated target phrases. The spotting of the incorrect target phrases and the paraphrasing of the source phrases can be automated. Promising oracle figures are obtained, validating the claim that some variations of the input sentence might be more easily translated than others by a given system. Paraphrases have also been used to represent alternative inputs encoded in lattices using existing (Schroeder et al., 2009) or automatically built paraphrases (Onishi et al., 2010; Du et al., 2010). In this scenario, paraphrases are in fact competing with each other, whereas in our proposal paraphrases *collectively participate in evaluating the quality of each translation for a source phrase*. We believe that if two phrases are indeed paraphrases in context, then their respective set of translations are both relevant to translate the two phrases. The target language model nevertheless still has an important role to play to select appropriate translations among semantically-compatible translations (i.e., target side paraphrases) in the specific context of a generated target hypothesis.

Lastly, automatic sentential paraphrasing has also been used in SMT to build alternative reference translations for parameter optimization (Madnani et al., 2008) and to build alternative training corpora (Bond et al., 2008).

3 Towards better exploitation of training corpora in phrase-based SMT

In typical phrase-based SMT settings (Koehn et al., 2003), words from the source side of the corpus are first aligned to words on the target side and biphrases are extracted from each training sentence using some heuristics on the word alignments. A source phrase f in a sentence being translated may therefore be aligned to a variety of target phrases. In the example on Figure 3, f is aligned some number of times in the training corpus to target phrases e_1, e_2, e_3 and e_5. Using the number of times f is paired with some target phrase e_i, $count(f, e_i)$, relative frequency estimation can be used to compute the probability of translation e_i given source phrase f:

$$p_{rel}(e_i|f) = \frac{count(f, e_i)}{\sum_j count(f, e_j)} \qquad (1)$$

This value, together with other estimates of how appropriate a translation pair (f, e_i) is, are recorded in a phrase table, which typically discards all contextual information.[8] Therefore, the translation distribution of some phrase is globally estimated from a training corpus independently of the actual context of that phrase.[9] On Figure 3, phrase f has at least two distinct senses: one represented by set $\mathcal{E}$, which in our example corresponds to the appropriate sense for a particular occurrence of f in a test sentence; and one which corresponds to translation e_5. A typical problem, due to the lack of context modeling,

[6] The default strategy for most decoders is to copy out-of-vocabulary tokens into the final text.

[7] Doing it in conjunction with our approach for improving the translation of known phrases is part of our future work.

[8] See (Carpuat and Wu, 2007; Stroppa et al., 2007; Max et al., 2008; Gimpel and Smith, 2008; Haque et al., 2009) for notable exceptions.

[9] Context is in fact taken into account to some extent by the target language model, which should score higher translations that are more appropriate given a target translation hypothesis being built. In fact, in this work we consider the target language model as the main source of information for selecting among acceptable target phrases (target language paraphrases).

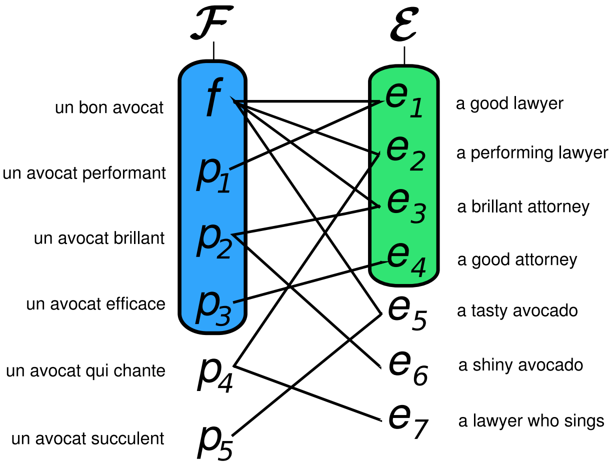


Figure 3: Example of possible source equivalents and translations for phrase occurrence f "*un bon avocat*" in the sentence "*L'embauche d'un bon avocat est cruciale quelle que soit l'activité*" ("*Hiring a good lawyer is crucial to any business"*). Set $\mathcal{E}$ represents target phrase types that are acceptable translations given the particular context of f, and set $\mathcal{F}$ represents source phrase types that can be in a paraphrasing relation to f depending on the context they appear in.

is that in situations such as $\forall e_i \in \mathcal{E}, count(f, e_5) \gg count(f, e_i)$, it is very unlikely that a correct translation will be selected during decoding against the incorrect but much more frequent one. Taking an extreme view on this issue, it is in fact desirable that when estimating phrase translation probabilities for a phrase f, translations of incompatible senses be not considered.[10] Of course, this raises the difficult issue of sense clustering of phrases. We propose here an intermediary solution, which consists in considering each occurrence in the training corpus as counting a number of times that depends on its contextual similarity with the occurrence of f from the test file, through the following additional translation model :

$$p_{cont}(e_i|f) = \frac{\sum_{\langle f_k, e_i \rangle} sim_{cont}(C(f), C(f_k))}{\sum_{\langle f_k, e_j \rangle} sim_{cont}(C(f), C(f_k))} \tag{2}$$

where f is some source phrase to translate and f_k an example of f in the training corpus, $\langle f_k, e_i \rangle$ is a biphrase from the training corpus, $C(f)$ the context of some source phrase, $C(f_k)$ the context of a particular example of f in the training corpus, sim_{phr} a function indicating the contextual similarity between two phrase contexts, and e_j is any possible translation of f.

The problem of modeling phrase translation is however not limited to inappropriate training examples. For various reasons, legitimate occurrences of source phrases may not be considered when building a phrase's translation distribution. We describe those cases by considering the possible source phrases p_i from Figure 3:

- p_1's only translation, e_1, is a common translation with f; each contextually-appropriate example of p_1 should reinforce the probability of e_1 being a translation for f.

- Contextually-appropriate examples of p_2 can reinforce e_3. Translation e_6 should correspond to contextually-inappropriate examples of p_2, so e_6 should not be considered as a new potential translation for f.

- Contrarily to the examples of p_2 translating as e_6, examples of p_3 translating as e_4 are much more likely of being contextually-appropriate with f, meaning that f could be substituted with most p_3 examples. Therefore, e_4, which was not initially considered as a possible translation of f, could now be considered as such.

- p_4 shares a translation with f, e_2, but this is due to the polysemous nature of this translation. Again, all examples of p_4 should be found contextually-inappropriate with f, and their translations should not be considered when estimating the translations of f.

- Lastly, the case of the common translation e_5 between f and p_5 illustrates a consequence of the polysemous nature of the source phrase corresponding to word sequence f: translations corresponding to other senses of f should not get reinforced by paraphrase examples such as those of p_5 as these examples should be found contextually-inappropriate with f.

[10] Put differently, is it more acceptable to copy a source word in the target hypothesis or to incorrectly translate it when the confidence about its being incorrect is high?

We build a separate translation model for translations estimated through paraphrases, defined as follows:

$$p_{para}(e_i|f) = \frac{\sum_{\langle p_k,e_i \rangle} sim_{para}(C(f), C(p_k))}{\sum_{\langle p_k,e_j \rangle} sim_{para}(C(f), C(p_k))} \quad (3)$$

where p_k is a paraphrase of f, $\langle p_k, e_i \rangle$ is a biphrase from the training corpus such that e_i is also a translation of f, $C(f)$ the context of a given source phrase for which we are estimating the translation distribution, $C(p_k)$ the context of a particular example of p_k in the training corpus, sim_{para} a function indicating the contextual similarity between a phrase context and a paraphrase context, and e_j is any possible translation of f.

Several requirements can be drawn from the previous description:

1. **List of potential paraphrases**: some mechanism for finding potential paraphrases for source phrases is required, and several such mechanisms could be combined. Pivoting via bilingual corpora, a natural strategy given the issue at hand, is just one among many different proposed strategies (Madnani and Dorr, 2010).

2. **Contextual similarity measure**: a similarity measure between the contexts of two phrases or two potential local paraphrases is required. This automatic measure should ideally be able to model not only syntactic but also semantic and pragmatic information.

3. **Robust translation evaluation**: our approach is designed to reinforce estimates for any contextually-appropriate translations of a phrase, as shown by set $\mathcal{E}$ on Figure 3. It is therefore important to have some means of accepting them as subparts of valid translations. Robustness in Machine Translation evaluation is an active domain, and potential candidates include using BLEU-like metrics with multiple references, Human-targeted Translation Error Rate (Snover et al., 2006) and the use of paraphrases for reference translations (Kauchak and Barzilay, 2006).

	train		dev.		test	
	# sent.	# tok.	# sent.	# tok.	# sent.	# tok.
en	318K	9.1M	500	14,0K	500	13,6K
fr	318K	10.3M	500	16,1K	500	15,7k

Figure 4: Statistics of the corpora used.

In this paper, we want to evaluate whether an endogenous approach for finding paraphrases can lead to some improvement in translation performance. Note that we will not consider in this initial work the possibility of adding new translations to phrases (such as e_4 for f on Figure 3) as it adds complexity and should be investigated when the other simpler cases can be handled successfully.

In the following section, we describe experiments in which the original bilingual corpus is the only resource used to find potential paraphrases and to estimate phrase translations in context. We chose a very simple measure of similarity, and let to our future work the task of improving context modeling. As regards evaluation, we will resort to various ways to measure the impact of our implementation on translation performance.

4 Experiments and results

4.1 Data and baseline SMT systems

We have conducted our experiments using the MOSES[11] package to build state-of-the-art phrase-based SMT systems for phrases of up to 5 tokens, using standard parameters and MERT for optimizing model weights. We used a subpart of the Europarl corpus[12] in French and English as our training corpus and built baseline MOSES systems (**bsl**) in both directions. The target side of the training corpus was used to train 3-gram target language model with modified Kneser-Ney smoothing. Held-out datasets were used for development and testing. The characteristics of all corpora are described in Figure 4.

4.2 Example-based Paraphrasing SMT systems

We also built systems that exploit phrase and paraphrase context under the form of two additional models p_{cont} and p_{para} described in section 3. These

[11] http://www.statmt.org/moses

[12] http://www.statmt.org/europarl

	phrase table size	num. entries
	en→fr	
baseline	240Mb	2.4M
our systems	5.0Gb	37.5M
	fr→en	
baseline	193Mb	1.9M
our systems	4.0Gb	30.2M

Figure 5: Statistics on the size and the number of entries of the phrase tables filtered on the development set.

models are added to the list of models used to evaluate the various translations of a phrase in the appropriate phrase tables, and are optimized with the other models by standard MERT.

In order to model context, we modified the source texts so that each phrase becomes unique in the phrase table, i.e. it has its own translation distribution. This is done (as in other works (Carpuat and Wu, 2007; Stroppa et al., 2007)) by transforming each token into a unique token, e.g. *token* → *token@337*. This therefore leads to a significant increase in the size of the phrase table, as illustrated on Figure 5, as all occurrences for the same phrase are not factored anymore.[13]

We chose a very simple initial definition of context similarity based on the presence of common *n*-grams in the immediate vicinity of two phrases. Let $length_{left}$ (resp. $length_{right}$) be the length of the longest common *n*-gram in the immediate vicinity on the left (resp. right) of two phrases in context ($C(f)$ and $C(f_i)$). For instance, given the two following contexts (phrases under focus are in bold and common *n*-grams are underlined):

1. *the commission accepts the substance of the* ***amendments@11257 proposed@11258 by@11259*** *the committee on fisheries ...*

2. *this is why we shall support all of the* ***amendments put forward by*** *the committee on agriculture and rural development ...*

$length_{left} = 2$ and $length_{right} = 3$. We further define $length$ as:

$$length = \begin{cases} length_{left} + length_{right} \\ \quad \text{if } length_{left} > 0 \\ \quad \text{and } length_{right} > 0 \\ 0 \text{ otherwise} \end{cases} \qquad (4)$$

We can now define the two similarity functions used in Equations 2 and 3 that we used for our experiments:

$$sim_{cont}(C(f), C(f_i)) \;=\; (1 + length)^{\alpha} \qquad (5)$$
$$sim_{para}(C(f), C(p_i)) \;=\; (length)^{\beta} \qquad (6)$$

The rationale for these functions is the following. Exact phrase examples add at least a translation count of 1, i.e. their translation is always taken into account to estimate p_{cont}. Paraphrase examples add a translation count of 0 if $length = 0$, i.e. their translation is not taken into account at all if surrounding *n*-gram similarity is too low. We used $\alpha = \beta = 1.5$. Algorithm 1 describes how the two models are estimated from the training data.

```
foreach phrase $f$ in training file do
    extract $C(f)$;
    /* phrase count */
    foreach unique phrase $f_i$ in $test(f)$) do
        extract $C(f_i)$;
        compute $sim_{cont}(C(f), C(f_i))$;
    end
    /* paraphrase count */
    foreach phrase $p_i$ in $para(f)$ do
        foreach unique phrase $f_j$ in $test(p_i)$ do
            extract $C(f_j)$;
            compute $sim_{para}(C(f), C(f_j))$;
        end
    end
end
estimate $p_{cont}$ and $_{para}$;
```

Algorithm 1: Model estimation for p_{cont} and p_{para}. Function $test(f)$ returns all unique phrases corresponding to phrase f from the test file. Function $para(f)$ return all phrases for which f is a known paraphrase.

We implemented the following strategy to find paraphrases for phrases in the test file. We extract all

[13] These volumes of data and our available hardware facilities for these experiments led us to initially limit the size of our data sets. We will discuss in section 5 how we intend to address this limitation in our future work.

	Left context	phrase/paraphrase	Right context
IS#1	*at the moment it is*	**up to each**	*member state to decide, and practice differs considerably from country to country*
PE#1	*... as regards the terminal portion in the cycle of nuclear fuel, it is*	**the responsability of each**	*member state to define its own policy .*
PT#1		la responsabilité de chaque	
IS#2	*that is why i find it*	**extremely regrettable that**	*the amendment on harmonising the re-registration of cars that have been involved in accidents ...*
PE#2	*for all these reasons and given your most excellent statement , i find it*	**a pity that**	*the new legal base for the daphne programme is so restrictive ...*
PT#2		dommage que	

Figure 6: Examples of paraphrases in context from the development file. The input sentence (IS) contains a source phrase of interest (in bold), the paraphrase example (PE) contains a paraphrase of that source phrase (in bold) for which a paraphrase translation (PT) is known.

paraphrases p for a phrase f by pivot: all target language phrases e aligned to f are first extracted, and all source language phrases p aligned to e are extracted. The following constraints are then applied to define which paraphrases are kept:

- string p is not included in string f and vice versa (in order to minimize the impact of alignment errors in the training corpus);
- the paraphrasing probability is greater than a fixed threshold: $para(f,p) \geq 10^{-2}$, where $para(f,p) = \sum_e p(e|f)p(p|e)$ (Bannard and Callison-Burch, 2005);
- the number of occurrences of phrase f and paraphrase p are equal or less than independent thresholds: $numOccs(f) \leq 100$ and $numOccs(p) \leq 1000$.[14]

Figure 6 shows examples of paraphrases in context with high similarity with some original phrase, and Figure 7 provides various statistics on the paraphrases extracted on the test file.

4.3 Results and analysis

Automatic evaluation results are reported in Table 8 for various configurations. We also wanted to focus our measures on content words, which are known to be important as regards information content in translation. We applied the contrastive lexical evaluation (CLE) methodology described in (Max et al., 2010), which indicates how many times source words grouped into user-defined classes were correctly translated or not across systems. These additional results are reported on Figure 9.

phrase	**# phrases**		**# paraphrased**		**# paraphrases**	
length	en	fr	en	fr	en	fr
1	13,620	15,707	458	725	1,824	2,684
2	13,120	15,207	4,127	4,481	18,871	19,700
3	12,620	14,707	4,782	5,715	24,111	27,377
4	12,120	14,208	2,859	4,078	15,071	20,345
5	11,623	13,711	1,171	2,275	6,077	12,132

Figure 7: Statistics on numbers of phrases, numbers of paraphrased phrases and numbers of paraphrases per phrase length.

On English to French translation, both additional features lead to improvements over the baseline with all metrics, including CLE, and their combination shows a strong improvement in TER (-1.55). CLE on content words reveals that the **para** feature seems particularly effective in reducing the number of words in all categories that only the baseline system translated correctly.

Results on French to English translation are less positive: neither **cont** nor **para** alone improve over the baseline with any metrics. However, their combination improves over the baseline with all metrics except BLEU, including a reduction of -1.07 in TER. Detailed analysis of CLE results shows that the translation of adjectives and nouns benefited

[14] The first threshold value was chosen as (Callison-Burch et al., 2005) report it to be an optimal sample size for estimating phrase translation probabilities. The relatively low value for the second threshold was selected to reduce computation time.

more from using our two additional models. Verbs, whose translation improved slightly, are strongly inflected in French, so finding examples for a given form is more difficult than for less inflected word categories, as is finding paraphrases with the appropriate inflection. Also, pivoting via English is one reason why paraphrases obtained via a low-inflected language can be of varying quality. Furthermore, the simplicity of our context modeling may have been ineffective in filtering out some bad examples. Overall, **para** was more effective with the low-inflected English as the source language, improving over the baseline with all metrics.

These results confirm that translation performance can be improved by exploiting context and paraphrases in the original training corpus only. We next attempted to measure whether some improvement in the quality of the paraphrases used would have some measurable impact on translation performance. To this end, we devised a semi-oracle experiment in the following way: the source and target test files were automatically aligned, and for each source phrase possible target phrases (i.e., reference translations) were extracted, and used as pivots to extract potential paraphrases, which were then filtered with the same constraints as previously. In this way, we exploit the information that paraphrases can at least produce the desired translation, but they may also propose other incorrect translations and/or be present in very few examples. Results appear in the **inf** rows of Tables 8 and 9. We obtain the most important improvement over the baseline in BLEU for the two language pairs (resp. +0.99 and +0.44), though the results for the other metrics for French to English translation are more difficult to interpret. For this language pair, possible reasons include again that the pivot language may not be appropriate, and also that the limitation to a single pivot[15] may not have produced more monolingual variation that might have proved useful. CLE on English to French, however, reveals significant gains with a relative improvement over the baseline of +116 content words. Under this condition, this result shows that the higher the quality of the paraphrases used, the more translation quality can be improved, which is in line with works that make use of human-made paraphrases to improve translation quality (Schroeder et al., 2009; Resnik et al., 2010).

	BLEU		NIST		TER		METEOR	
	en→fr							
bsl	30.28	-	6.66	-	57.86	-	54.79	-
+cont	**31.11**	+0.83	6.77	+0.11	57.24	-0.62	55.22	+0.43
+para	30.97	+0.69	6.74	+0.08	57.38	-0.48	**55.39**	+0.60
all	30.93	+0.65	**6.84**	+0.18	**56.31**	-1.55	55.28	+0.49
inf	31.27	+0.99	6.78	+0.12	57.22	-0.64	55.80	+1.01
	fr→en							
bsl	**29.90**	-	6.90	-	54.64	-	61.36	-
+cont	29.56	-0.34	6.89	-0.01	54.95	+0.31	60.98	-0.38
+para	29.70	-0.20	6.92	+0.02	54.64	+0.00	61.10	-0.26
all	29.75	-0.15	**7.03**	+0.13	**53.57**	-1.07	**61.63**	+0.27
inf	30.34	+0.44	6.93	+0.03	54.90	+0.26	60.99	-0.37

Figure 8: Automatic scores for the MOSES baseline systems (**bsl**), systems additionnally using the contextual feature (**+cont**), systems additionnally using the paraphrasing feature (**+para**), systems using both features (**all**), and pivot-informed systems (**inf**).

		Adj	Adv	Noun	Verb	$\sum$	
		en→fr					
+cont	-	74	28	113	60	275	
	+	55	35	114	85	289	+14
+para	-	62	12	82	46	202	
	+	58	32	111	78	279	+77
all	-	72	25	91	72	260	
	+	50	37	118	97	302	+42
inf	-	58	20	108	56	242	
	+	65	43	147	103	358	+116
		fr→en					
+cont	-	30	16	80	69	195	
	+	15	21	69	46	151	-44
+para	-	32	19	72	60	183	
	+	12	18	65	43	138	-45
all	-	21	18	67	61	167	
	+	30	18	94	48	190	+23
inf	-	38	21	83	66	208	
	+	31	23	106	57	217	+9

Figure 9: Contrastive lexical evaluation results per part-of-speech measured on the test file. '-' (resp. '+') rows indicate the number of source words that only **bsl** (resp. the compared system) correctly translated.

Table 10 presents a typology of paraphrases found in our development set and classifies the impact of using them for phrase translation estimation. As can be seen, more work is needed to better understand the characteristics of the phrases that should be paraphrased and of their paraphrases.

[15] Several pivot phrases may in fact have been automatically extracted for a given phrase, some of which being possible bad candidates.

Type	Impact	Examples
Morphological variants	+/-	(yugoslav republic ↔ yugoslavian republic), (go far ↔ goes far)
Synonymy	+	(duties ↔ obligations), (to look into ↔ to study)
Grammatical word substitution	?/-	(states in the ↔ the states of the), (amendments by ↔ amendments to)
Word deletion or insertion	?/-	(first reading, the → first reading the), (amendments by ↔ amendments proposed by)
Syntactic rewritings	+	(approval of the majority ↔ majority support), (capacity of the european union ↔ european union's ability)
Phrasal idiomatic substitutions	+	(must be said that the ↔ goes without saying that the), (is fully in line ↔ is totally coherent), (is amazing ↔ strikes me)
Context-dependent substitutions	+/-	(is not right ↔ is unacceptable), (offer my ↔ express my)
Alignment and translation problems	-	(unnecessary if ↔ vital if), (the crime ↔ organized), (ill-advised ↔ wise), (to begin by thanking ↔ to begin by congratulating)

Figure 10: Main types of paraphrase pairs found in our dev. and training corpora. Pairs shown have $length > 0$.

5 Conclusion and future work

We have introduced an original way of exploiting both context and paraphrasing for the estimation of phrase translations in phrase-based SMT. To our knowledge, this is the first time that paraphrases acquired in an endogenous manner have been shown to improve translation performance, which shows that bilingual corpora can be better exploited than they typically are. Our experiments further showed the promises of our approach when paraphrases of higher quality are available.

In the light of our results and our initial typology of paraphrases presented on Figure 10, as well as previous work on paraphrasing for SMT, the difficult question of what units should be paraphrased for what success should be addressed, taking into account parameters such as language pairs, quantity of training data and availability of external resources.

Our future work includes three main areas: first, we want to improve the modeling of context, by notably working on techniques inspired from Information Retrieval to quickly access contextually-similar examples of source phrases in bilingual corpora. Such *contextual sampling* on large bilingual corpora for phrases and their paraphrases, which could integrate more complex linguistic information, will allow us to assess our approach on more challenging conditions. This would also allow us to build contextual models on-the-fly, and experiment with using lattices to encode contextually estimated paraphrases. Second, we will combine paraphrases obtained via different techniques and resources, which will allow us to also learn translation distributions for phrases absent from the original corpus. Lastly, we want to also exploit paraphrases for the *additional* translations that they propose (such as e_4 on Figure 3) and that would be contextually similar in the target language to other existing translations of a given phrase or that could even represent a new sense of the original phrase.

Acknowledgements

This work was partly supported by ANR project Trace (ANR-09-CORD-023). The author would like to thank the anonymous reviewers for their helpful questions and comments.

References

Sadaf Abdul-Rauf and Holger Schwenk. 2009. On the Use of Comparable Corpora to Improve SMT performance. In *Proceedings of EACL*, Athens, Greece.

Wilker Aziz, Marc Dymetman, Shachar Mirkin, Lucia Specia, Nicola Cancedda, and Ido Dagan. 2010. Learning an Expert from Human Annotations in Statistical Machine Translation: the Case of Out-of-Vocabulary Words. In *Proceedings of EAMT*, Saint-Raphael, France.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with Bilingual Parallel Corpora. In *Proceedings of ACL*, Ann Arbor, USA.

Francis Bond, Eric Nichols, Darren Scott Appling, and Michael Paul. 2008. Improving statistical machine translation by paraphrasing the training data. In *Proceedings of IWSLT*, Hawai, USA.

Chris Callison-Burch, Colin Bannard, and Josh Schroeder. 2005. Scaling Phrase-Based Statistical Machine Translation to Larger Corpora and Longer Phrases. In *Proceedings of ACL*, Ann Arbor, USA.

Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved Statistical Machine Translation Using Paraphrases. In *Proceedings of NAACL*, New York, USA.

Chris Callison-Burch. 2008. Syntactic Constraints on Paraphrases Extracted from Parallel Corpora. In *Proceedings of EMNLP*, Hawai, USA.

Marine Carpuat and Dekai Wu. 2007. Context-Dependent Phrasal Translation Lexicons for Statistical Machine Translation. In *Proceedings of Machine Translation Summit XI*, Copenhagen, Denmark.

Marine Carpuat. 2009. One Translation Per Discourse. In *Proceedings of the NAACL-HLT Workshop on Semantic Evaluations*, Boulder, USA.

Jinhua Du, Jie Jiang, and Andy Way. 2010. Facilitating Translation Using Source Language Paraphrase Lattices. In *Proceedings of EMNLP*, Cambridge, USA.

Kevin Gimpel and Noah A. Smith. 2008. Rich Source-Side Context for Statistical Machine Translation. In *Proceedings of the ACL Workshop on Statistical Machine Translation*, Columbus, USA.

Rejwanul Haque, Sudip Kumar Naskar, Yanjun Ma, and Andy Way. 2009. Using Supertags as Source Language Context in SMT. In *Proceedings of EAMT*, Barcelona, Spain.

Almut Silja Hildebrand, Matthias Eck, Stephan Vogel, and Alex Waibel. 2005. Adaptation of the Translation Model for Statistical Machine Translation Based on Information Retrieval. In *Proceedings of EAMT*, Budapest, Hungary.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for Automatic Evaluation. In *Proceedings of NAACL HLT*, New York, USA.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of NAACL HLT*, Edmonton, Canada.

Stanley Kok and Chris Brockett. 2010. Hitting the Right Paraphrases in Good Time. In *Proceedings of NAACL*, Los Angeles, USA.

Adam Lopez. 2008. Tera-Scale Translation Models via Pattern Matching. In *Proceedings of COLING*, Manchester, UK.

Nitin Madnani and Bonnie J. Dorr. 2010. Generating Phrasal & Sentential Paraphrases: A Survey of Data-Driven Methods. *Computational Linguistics*, 36(3).

Nitin Madnani, Philip Resnik, Bonnie J. Dorr, and Richard Schwartz. 2008. Are Multiple Reference Translations Necessary? Investigating the Value of Paraphrased Reference Translations in Parameter Optimization. In *Proceedings of AMTA*, Waikiki, USA.

Yuval Marton, Chris Callison-Burch, and Philip Resnik. 2009. Improved Statistical Machine Translation Using Monolingually-derived Paraphrases. In *Proceedings of EMNLP*, Singapore.

Aurélien Max, Rafik Makhloufi, and Philippe Langlais. 2008. Explorations in using grammatical dependencies for contextual phrase translation disambiguation. In *Proceedings of EAMT*, Hamburg, Germany.

Aurélien Max, Josep M. Crego, and François Yvon. 2010. Contrastive Lexical Evaluation of Machine Translation. In *Proceedings of LREC*, Valletta, Malta.

Aurélien Max. 2008. Local rephrasing suggestions for supporting the work of writers. In *Proceedings of GoTAL*, Gothenburg, Sweden.

Shachar Mirkin, Lucia Specia, Nicola Cancedda, Ido Dagan, Marc Dymetman, and Idan Szpektor. 2009. Source-Language Entailment Modeling for Translating Unknown Terms. In *Proceedings of ACL*, Singapore.

Behrang Mohit and Rebecca Hwa. 2007. Localization of Difficult-to-Translate Phrases. In *Proceedings of the ACL Workshop on Statistical Machine Translation*, Prague, Czech Republic.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving Machine Translation Performance by Exploiting Non-parallel Corpora. *Computational Linguistics*, 31(4).

Takashi Onishi, Masao Utiyama, and Eiichiro Sumita. 2010. Paraphrase Lattice for Statistical Machine Translation. In *Proceedings of ACL, short paper session*, Uppsala, Sweden.

Philip Resnik, Olivia Buzek, Chang Hu, Yakov Kronrod, Alex Quinn, and Benjamin B. Bederson. 2010. Improving Translation via Targeted Paraphrasing. In *Proceedings of EMNLP*, Cambridge, USA.

Josh Schroeder, Trevor Cohn, and Philipp Koehn. 2009. Word Lattices for Multi-Source Translation. In *Proceedings of EACL*, Athens, Greece.

Matthew Snover, Bonnie J. Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of AMTA*, Boston, USA.

Nicolas Stroppa, Antal van den Bosch, and Andy Way. 2007. Exploiting Source Similarity for SMT using Context-Informed Features. In *Proceedings of TMI*, Skovde, Sweden.

Guillaume Wisniewski, Alexandre Allauzen, and François Yvon. 2010. Assessing Phrase-based Translation Models with Oracle Decoding. In *Proceedings of EMNLP*, Cambridge, USA.

Combining Unsupervised and Supervised Alignments for MT: An Empirical Study

Jinxi Xu and **Antti-Veikko I. Rosti**
Raytheon BBN Technologies, 10 Moulton Street, Cambridge, MA 02138, USA
{jxu,arosti}@bbn.com

Abstract

Word alignment plays a central role in statistical MT (SMT) since almost all SMT systems extract translation rules from word aligned parallel training data. While most SMT systems use unsupervised algorithms (e.g. GIZA++) for training word alignment, supervised methods, which exploit a small amount of human-aligned data, have become increasingly popular recently. This work empirically studies the performance of these two classes of alignment algorithms and explores strategies to combine them to improve overall system performance. We used two unsupervised aligners, GIZA++ and HMM, and one supervised aligner, ITG, in this study. To avoid language and genre specific conclusions, we ran experiments on test sets consisting of two language pairs (Chinese-to-English and Arabic-to-English) and two genres (newswire and weblog). Results show that the two classes of algorithms achieve the same level of MT performance. Modest improvements were achieved by taking the union of the translation grammars extracted from different alignments. Significant improvements (around 1.0 in BLEU) were achieved by combining outputs of different systems trained with different alignments. The improvements are consistent across languages and genres.

1 Introduction

Word alignment plays a central role in training statistical machine translation (SMT) systems since almost all SMT systems extract translation rules from word aligned parallel training data. Until recently, most SMT systems used GIZA++ (Och and Ney, 2003), an unsupervised algorithm, for aligning parallel training data. In recent years, with the availability of human aligned training data, supervised methods (e.g. the ITG aligner (Haghighi et al., 2009)) have become increasingly popular.

The main objective of this work is to show the two classes (unsupervised and supervised) of algorithms are complementary and combining them will improve overall system performance. The use of human aligned training data allows supervised methods such as ITG to more accurately align frequent words, such as the alignments of Chinese particles (e.g. “bei”, “de”, etc) to their English equivalents (e.g. “is/are/was/..”, “of”, etc). On the other hand, supervised methods can be affected by suboptimal alignments in hand-aligned data. For example, the hand-aligned data used in our experiments contain some coarse-grained alignments (e.g. “lianhe guo” to “United Nations”) although fine-grained alignments (“lian-he” to “United” and “guo” to “Nations”) are usually more appropriate for SMT. Unsupervised methods are less likely to be affected by this problem. We used two well studied unsupervised aligners, GIZA++ (Och and Ney, 2003) and HMM (Liang et al., 2006) and one supervised aligner, ITG (Haghighi et al., 2009) as representatives in this work.

We explored two techniques to combine different alignment algorithms. One is to take the union of the translation rules extracted from alignments produced by different aligners. This is motivated by studies that showed that the coverage of translation rules is critical to SMT (DeNeefe et al., 2007). The

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 667–673,
MIT, Massachusetts, USA, 9-11 October 2010.

other method is to combine the outputs of different MT systems trained using different aligners. Assuming different systems make independent errors, system combination can generate a better translation than those of individual systems through voting (Rosti et al., 2007).

Our work differs from previous work in two ways. Past studies of combining alternative alignments focused on minimizing alignment errors, usually by merging alternative alignments for a sentence pair into a single alignment with the fewest number of incorrect alignment links (Ayan and Dorr, 2006). In contrast, our work is based on the assumption that perfect word alignment is impossible due to the intrinsic difficulty of the problem, and it is more effective to resolve translation ambiguities at later stages of the MT pipeline. A main focus of much previous work on word alignments is on theoretical aspects of the proposed algorithms. In contrast, the nature of this work is purely empirical. Our system was trained on a large amount of training data and evaluated on multiple languages (Chinese-to-English and Arabic-to-English) and multiple genres (newswire and weblog). Furthermore, we used a state of the art string-to-tree decoder (Shen et al., 2008) to establish the strongest possible baseline. In comparison, experiments in previous studies typically used one language pair and one genre (usually newswire), a reduced amount of training data and a phrase based decoder.

This paper is organized as follows. Section 2 describes the three alignment algorithms. Section 3 describes the two methods used to combine these aligners to improve MT. The experimental setup used to compare these methods is presented in Section 4. Section 5 shows the results including a discussion. Section 6 discusses related work. Section 7 concludes the paper.

2 Alignment Algorithms

We used three aligners in this work: GIZA++ (Och and Ney, 2003), jointly trained HMM (Liang et al., 2006), and ITG (Haghighi et al., 2009). GIZA++ is an unsupervised method based on models 1-5 of Brown et al. (1993). Given a sentence pair $e - f$, it seeks the alignment a that maximizes the probability $P(f, a|e)$. As in most previous studies using GIZA++, we ran GIZA++ in both directions, from e to f and from f to e, and symmetrized the bidirectional alignments into one, using a method similar to the grow-diagonal-final method described in Och and Ney (2003). We ran GIZA++ up to model 4.

The jointly trained HMM aligner, or HMM for short, is also unsupervised but it uses a small amount of hand-aligned data to tweak a few high level parameters. Low level parameters are estimated in an unsupervised manner like GIZA++.

The ITG aligner is a supervised method whose parameters are tuned to optimize alignment accuracy on hand-aligned data. It uses the inversion transduction grammar (ITG) (Wu, 1997) to narrow the space of possible alignments. Since the ITG aligner uses features extracted from HMM alignments, HMM was run as a prepossessing step in our experiments. Both the HMM and ITG aligners are publicly available[1].

3 Methods of Combining Alternative Alignments for MT

We explored two methods of combining alternative alignments for MT. One is to extract translation rules from the three alternative alignments and take the union of the three sets of rules as the single translation grammar. Procedurally, this is done by concatenating the alignment files before extracting translation rules. We call this method *unioned grammar*. This method greatly increases the coverage of the rules, as the unioned translation grammar has about 80% more rules than the ones extracted from the individual alignment in our experiments. As such, decoding is also slower.

The other is to use system combination to combine outputs of systems trained using different aligners. Due to differences in the alignment algorithms, these systems would produce different hypotheses with independent errors. Combining a diverse set of hypotheses could improve overall system performance. While system combination is a well-known technique, to our knowledge this work is the first to apply it to explicitly exploit complementary alignment algorithms on a large scale.

Since system combination is an established technique, here we only briefly discuss our system com-

[1] http://code.google.com/p/berkeleyaligner/

bination setup. The basic algorithm was described in Rosti et al. (2007). In this work, we use incremental hypothesis alignment with flexible matching (Rosti et al., 2009) to produce the confusion networks. 10-best lists from all systems are collected first. All 1-best hypotheses for each segment are used as confusion network skeletons, the remaining hypotheses are aligned to the confusion networks, and the resulting networks are connected in parallel into a joint lattice with skeleton specific prior probabilities estimated from the alignment statistics on the initial arcs. This lattice is expanded with an unpruned bigram language model and the system combination weights are tuned directly to maximize the BLEU score of the 1-best decoding outputs. Given the tuned system combination weights, a 300-best list is extracted from the lattice, the hypotheses are rescored using an unpruned 5-gram language model, and a second set of system combination weights is tuned to maximize the BLEU score of the 1-best hypothesis of the re-scored 300-best list. The same rescoring step is also applied to the outputs of individual systems.

4 Experiment Setup

To establish strong baselines, we used a string-to-tree SMT system (Shen et al., 2008), one of the top performing systems in the NIST 2009 MT evaluation, and trained it with very large amounts of parallel and language model data. The system used large sets of discriminatively tuned features (up to 55,000 on Arabic) inspired by the work of Chiang et al. (2009). To avoid drawing language, genre, and metric specific conclusions, we experimented with two language pairs, Arabic-English and Chinese-English, and two genres, newswire and weblog, and report both BLEU (Papineni et al., 2002) and TER (Snover et al., 2006) scores. Systems were tuned to maximize BLEU on the tuning set using a procedure described in Devlin (2009).

The sizes of the parallel training corpora are 238M words (target side) for Arabic-English MT and 265M words for Chinese-English. While the majority of the data is publicly available from the Linguistic Data Consortium (LDC), some of the data is available under the DARPA GALE program. Due to the size of the parallel corpora, we divided them into five chunks and aligned them in parallel to save time. Due to its running complexity, we ran ITG only on sentences with 60 or fewer words. For longer sentences, we used HMM alignments instead, which were conveniently generated in the preprocessing step of ITG aligner. For language model training, we used about 9 billion words of English text, most of which are from English Gigaword corpus and GoogleNews. Each system used a 3-gram LM for decoding and a 5-gram LM for re-scoring. The same 5-gram LM was also used for re-scoring system combination results.

For each combination of language pair and genre, we used three development sets:

- `Tune`, which was used to tune parameters of individual MT systems. Each system was tuned ten iterations based on BLEU.
- `SysCombTune`, which was used to tune parameters of system combination. A subset of it was also used as validation for determining the best iteration in tuning individual systems.
- `Test`, which was the blind test corpus for measuring performances of both individual systems and system combination.

Test materials were drawn from two sources: NIST MT evaluations 2004 to 2008, and development and evaluation data for the DARPA GALE program. Due to the mixing of different data sources, some test sentences have four reference translations while the rest have only one. The average number of references per test sentence varies across test sets. For this reason, MT scores are not comparable across test sets. Table 1 shows the size and the average number of references per sentence of the test sets.

Two hand-aligned corpora were used to train the ITG aligner: LDC2009E82 (Arabic-English) and LDC2009E83 (Chinese-English). We re-tokenized the corpora using our tokenizers and projected the LDC alignments to our tokenization heuristically. The projection was not perfect and sometimes created very coarse-grained alignments. We used a set of filters to remove such problematic data. We ended up with 3,667 Arabic-English and 879 Chinese-English hand-aligned sentence pairs with sufficient quality for training automatic aligners.

language and genre	Tune	SysCombTune	Test
Arabic newswire	2963 (2.9)	3223 (2.7)	2242 (2.7)
Arabic web	4597 (1.5)	4526 (1.4)	2703 (2.7)
Chinese newswire	3085 (2.6)	3001 (2.7)	2055 (1.4)
Chinese web	4221 (1.3)	4285 (1.3)	3092 (1.2)

Table 1: Numbers of sentences and average number of references (in parentheses) of test sets

5 Results

Three baseline systems were trained using the three different aligners. Case insensitive BLEU and TER scores for Arabic newswire, Arabic weblog, Chinese newswire, and Chinese weblog are shown in Tables 2, 3, 4, and 5, respectively[2]. The BLEU scores on the `Test` set are fairly similar but the ordering between different alignment algorithms is mixed between different languages and genres. To compare the two alignment combination strategies, we trained a system using the union of the rules extracted from the alternative alignments (`union` in the tables) and a combination of the three baseline system outputs (`3 syscomb` in the tables). The system with the unioned grammar was also added as an additional system in the combination marked by `4 syscomb`.

As seen in the tables, unioned grammar and system combination improve MT on both languages (Arabic and Chinese) and both genres (newswire and weblog). While there are improvements on both `SysCombTune` and `Test`, the results on `SysCombTune` are not totally fair since it was used for tuning system combination weights and as validation for optimizing weights of the MT systems. Therefore our discussion will focus on results on `Test`. (We did not show scores on `Tune` because systems were directly tuned on it.) Statistical significance is determined at 95% confidence level using the bootstrap method described in Koehn (2004), and is only applied on results obtained on the blind `Test` set.

For unioned grammar, the overall improvement in BLEU is modest, ranging from 0.1 to 0.6 point compared with the best baseline system, with little change in TER score. The improvements in BLEU score are statistically significant for Arabic (both genres), but not for Chinese. The improvements in TER are not significant for either language.

System combination produces bigger improvements in performance. Compared with the best baseline system, the improvement in BLEU ranges from 0.8 to 1.6 point. There are also noticeable improvements in TER, around 1.0 point. The TER improvements are mostly explained by the hypothesis alignment algorithm which is closely related to TER scoring (Rosti et al., 2009). The results are interesting because all three baseline systems (GIZA++, HMM and ITG) are identical except for the word alignments used in rule extraction. The results confirm that the aligners are indeed complementary, as we conjectured earlier. Also, the four-system combination yields consistent gains over the three-system combination, suggesting that the system using the unioned grammar is somewhat complementary to the three baseline systems. The statistical test indicates that both the three and four system combinations are significantly better than the single best alignment system for all languages and genres in BLEU and TER. In most cases, they are also significantly better than unioned grammar.

Somewhat surprisingly, the GIZA++ trained system is slightly better than the ITG trained system on all genres but Chinese weblog. However, we should point out that such a comparison is not entirely fair. First, we only ran ITG on short sentences. (For long sentences, we had to settle for HMM alignments for computing reasons.) Second, the hand-aligned data used for ITG training are not very clean, as we said before. The ITG results could be improved if these problems were not present.

[2] Dagger (†) indicates statistically better results than the best individual alignment system. Double dagger (‡) indicates statistically better results than both best individual alignment and unioned grammar. Bold indicates best Test set performance among individual alignment systems.

	SysCombTune		Test	
System	BLEU	TER	BLEU	TER
GIZA++	51.31	38.01	**50.96**	**38.38**
HMM	50.87	38.49	50.84	38.87
ITG	51.04	38.44	50.69	38.94
union	51.55	37.93	51.53†	38.32
3 syscomb	52.66	37.20	52.43‡	37.69‡
4 syscomb	52.80	37.05	52.55‡	37.46‡

Table 2: MT results on Arabic newswire (see footnote 2).

	SysCombTune		Test	
System	BLEU	TER	BLEU	TER
GIZA++	27.49	55.00	**38.00**	**49.55**
HMM	27.42	55.53	37.81	50.12
ITG	27.19	55.32	37.77	49.94
union	27.66	54.82	38.43†	49.43
3 syscomb	27.65	53.89	38.70†	48.72‡
4 syscomb	27.83	53.68	38.82‡	48.53‡

Table 3: MT results on Arabic weblog (see footnote 2).

	SysCombTune		Test	
System	BLEU	TER	BLEU	TER
GIZA++	36.42	54.21	**26.77**	57.67
HMM	36.12	54.50	26.17	58.22
ITG	36.23	54.11	26.53	**57.40**
union	36.57	54.07	26.83	57.37
3 syscomb	37.60	53.19	27.46‡	56.88‡
4 syscomb	37.77	53.11	27.57‡	56.57‡

Table 4: MT results on Chinese newswire (see footnote 2).

	SysCombTune		Test	
System	BLEU	TER	BLEU	TER
GIZA++	18.71	64.10	16.94	63.46
HMM	18.35	64.66	16.66	64.02
ITG	18.76	63.67	**16.97**	**63.29**
union	18.97	63.86	17.22	63.20
3 syscomb	19.66	63.40	17.98‡	62.47‡
4 syscomb	19.80	63.32	18.05‡	62.36‡

Table 5: MT results on Chinese weblog (see footnote 2).

5.1 Discussion

Inter-aligner agreements provide additional evidence about the differences between the aligners. Suppose on a common data set, the sets of alignment links produced by two aligners are A and B, we compute their agreement as $(|A \bigcap B|/|A| + |A \bigcap B|/|B|)/2$. (This is the average of recall and precision of one set by treating the other set as reference.) The agreement between GIZA++ and ITG is around 78% on a subset of the Arabic-English parallel data. The agreements between GIZA++ and HMM, and between HMM and ITG are slightly higher, around 83%. Since ITG could not align long sentences, we only used short sentences (at most 60 words in length) in our calculation.

Due to the large differences between the aligners, significantly more rules were extracted with the unioned grammar method in our experiments. On average, the size of the grammar (number of rules) was increased by about 80% compared with the baseline systems. The larger grammar results in more combinations of partial theories in decoding. However, for computing reasons, we kept the beam size of the decoder constant despite the increase in grammar size, potentially pruning out good theories. Performance could be improved further if larger beam sizes were used. We will leave this to future work.

6 Related Work

Ayan and Dorr (2006) described a method to minimize alignment errors by combining alternative alignments into a single alignment for each sentence pair. Deng and Zhou (2009) used the number of extractable translation pairs as the objective function for alignment combination. Och and Ney (2003) and Koehn et al. (2003) used heuristics to merge the bidirectional GIZA++ alignments into a single alignment. Despite differences in algorithms and objective functions in these studies, they all attempted to produce a single final alignment for each sentence pair. In comparison, all alternative alignments are directly used by the translation system in this work.

The unioned grammar method in this work is very similar to Giménez and Màrquez (2005), which combined phrase pairs extracted from different alignments into a single phrase table. The difference from that work is that our focus is to leverage complementary alignment algorithms, while theirs was to leverage alignments of different lexical units produced by the same aligner.

Some studies leveraged other types of differences between systems to improve MT. For example, de Gispert et al. (2009) combined systems trained with different tokenizations.

The theory behind the GIZA++ aligner was due to Brown et al. (1993). The theory of Inversion Transduction Grammars (ITG) was due to Wu (1997). The ITG aligner (Haghighi et al., 2009) used in this work extended the original ITG to handle blocks of words in addition to single words. The use of HMM for word alignment can be traced as far back as to Vogel et al. (1996). The HMM aligner used in this work was due to Liang et al. (2006). It refined the original HMM alignment algorithm by jointly training two HMMs, one in each direction. Furthermore, it used a small amount of supervised data to tweak some high level parameters, although it did not directly use the supervised data in training.

7 Conclusions

We explored two methods to exploit complementary alignment algorithms. One is to extract translation rules from all alternative alignments. The other is to combine outputs of different MT systems trained using different aligners. Experiments on two language pairs and two genres show consistent improvements over the baseline systems.

Acknowledgments

This work was supported by DARPA/IPTO Contract No. HR0011-06-C-0022 under the GALE program[3] (Approved for Public Release, Distribution Unlimited). The authors are grateful to John DeNero and John Blitzer for their help with the Berkeley HMM and ITG aligners.

[3] The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

References

Necip Fazil Ayan and Bonnie J. Dorr. 2006. A maximum entropy approach to combining word alignments. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL*, pages 96–103.

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.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL*, pages 218–226.

Adrià de Gispert, Sami Virpioja, Mikko Kurimo, and William Byrne. 2009. Minimum Bayes risk combination of translation hypotheses from alternative morphological decompositions. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL*, pages 73–76.

Steve DeNeefe, Kevin Knight, Wei Wang, and Daniel Marcu. 2007. What can syntax-based MT learn from phrase-based MT? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 755–763.

Yonggang Deng and Bowen Zhou. 2009. Optimizing word alignment combination for phrase table training. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, pages 229–232.

Jacob Devlin. 2009. Lexical features for statistical machine translation. Master's thesis, University of Maryland.

Jesús Giménez and Lluís Màrquez. 2005. Combining linguistic data views for phrase-based SMT. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 145–148.

Aria Haghighi, John Blitzer, John DeNero, and Dan Klein. 2009. Better word alignments with supervised ITG models. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, pages 923–931.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the ACL*, pages 48–54.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL*, pages 104–111.

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

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*, pages 311–318.

Antti-Veikko I. Rosti, Bing Xiang, Spyros Matsoukas, Richard Schwartz, Necip Fazil Ayan, and Bonnie J. Dorr. 2007. Combining outputs from multiple machine translation systems. In *Proceedings of Human Language Technologies 2007: The Conference of the North American Chapter of the ACL*, pages 228–235.

Antti-Veikko I. Rosti, Bing Zhang, Spyros Matsoukas, and Richard Schwartz. 2009. Incremental hypothesis alignment with flexible matching for building confusion networks: BBN system description for WMT09 system combination task. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 61–65.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-08: HLT*, pages 577–585.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciula, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*, pages 223–231.

Stephan Vogel, Hermann Ney, and Christoph Tillman. 1996. HMM-based word alignment in statistical translation. In *The 16th International Conference on Computational Linguistics*, pages 836–841.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3).

Top-Down Nearly-Context-Sensitive Parsing

Eugene Charniak
Brown Laboratory for Linguistic Information Processing (BLLIP)
Brown University, Providence, RI 02912
ec@cs.brown.edu

Abstract

We present a new syntactic parser that works left-to-right and top down, thus maintaining a fully-connected parse tree for a few alternative parse hypotheses. All of the commonly used statistical parsers use context-free dynamic programming algorithms and as such work bottom up on the entire sentence. Thus they only find a complete fully connected parse at the very end. In contrast, both subjective and experimental evidence show that people understand a sentence word-to-word as they go along, or close to it. The constraint that the parser keeps one or more fully connected syntactic trees is intended to operationalize this cognitive fact. Our parser achieves a new best result for top-down parsers of 89.4%,a 20% error reduction over the previous single-parser best result for parsers of this type of 86.8% (Roark, 2001). The improved performance is due to embracing the very large feature set available in exchange for giving up dynamic programming.

1 Introduction

We present a new syntactic parser that works top-down and left-to-right, maintaining a fully-connected parse tree for a few alternative parse hypotheses. It is a Penn treebank (Marcus et al., 1993) parser in that it is capable of parsing the Penn treebank test sets, and is trained on the now standard training set. It achieves a new best result for this parser type.

All of the commonly used statistical parsers available on the web such as the Collins(/Bikel) (Collins, 2003) Charniak-Johnson(Charniak and Johnson, 2005), and Petrov-Klein (Petrov et al., 2006), parsers use context-free dynamic programming algorithms so they work bottom up on the entire sentence. Thus they only find a complete fully-connected parse at the very end.

In contrast human syntactic parsing must be fully connected (or close to it) as people are able to apply vast amounts of real-world knowledge to the process as it proceeds from word-to-word(van Gompel and Pickering, 2007). Thus any parser claiming cognitive plausibility must, to a first approximation, work in this left-to-right top-down fashion.

Our parser obtains a new best result for top-down parsers of 89.4% (on section 23 of the Penn Treebank). This is a 20% error reduction over the previous best single-parser result of 86.8%, achieved by Rork(Roark, 2001).

Our model is in the tradition of this latter parser. The current work's superior performance is not due to any innovation in architecture but in how probability distributions are computed. It differs from Roark in its explicit recognition that by giving up context-free dynamic programming we may embrace near context sensitivity and condition on many diverse pieces of information. (It is only "near" because we still only condition on a finite amount of information.) This is made possible by use of random-forests (Amit and Geman, 1997; Breiman, 2004; Xu and Jelinek, 2004) to choose features, provide smoothing, and finally do the probability computation. To the best of our knowledge ours is the first application of random-forests to parsing.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 674–683,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

Section two describes previous work on this type of parser, and in particular gives details on the Roark architecture we use. Section three describes how random forests allow us to integrate the diverse information sources that context-sensitive parsing allows. Section four gives implementation details. Section five is devoted to the main experimental finding of the paper along with subsidiary results showing the effects of the large feature set we now may use. Finally, section six suggests that because this parser type is comparatively little explored one may hope for further substantial improvements, and proposes avenues to be explored.

2 Previous Work on Top-Down Parsing and the Roark Model

We care about top-down incremental parsing because it automatically satisfies the criteria we have established for cognitive plausibility. Before looking at previous work on this type model we briefly discuss work that does *not* meet the criteria we have set out, but which people often assume does so.

We are using the terms "top-down" and "left-to-right" following e.g., (Abney and Johnson, 1991; Roark, 2001). In particular

> In top-down strategies a node is enumerated before any of its descendents.(Abney and Johnson, 1991)

In this era of statistical parsers it is useful to think in terms of possible conditioning information. In typical bottom up CKY parsing when creating, say, a constituent X from positions i to j we may not condition on its parent. That the grammar is "context-free" means that this constituent may be used anywhere.

Using our definition, the Earley parsing algorithm(Earley, 1970), which is often cited as "top-down," is no such thing. In fact, it is long been noted that the Earley algorithm is "almost identical"(Graham et al., 1980) to CKY. Again, when Earley posits an X it may not condition on the parent.

Similarly, consider the more recent work of Nivre(Nivre, 2003) and Henderson(Henderson, 2003). The reason these are not fully-connected is the same. While they are incremental parsers, they are not top down — both are shift-reduce parsers. Consider a constituency shift-reduce mechanism. Suppose we have a context-free rule S → NP VP. As we go left-to-right various terminal and non-terminals are added to and removed from the stack until at some point the top two items are NP and VP. Then a reduce operation replaces them with S. Note that this means that *none* of the words or sub-constituents of either the NP or VP are integrated into a single overall tree until the very end. This is clearly not fully connected. Since Nivre's parser is a dependency parser this exact case does not apply (as it does not use CFG rules), but similar situations arise. In particular, whenever a word is dependent on a word that appears later in the string, it remains unconnected on the stack until the second word appears. Naturally this is transitive so that the parser can, and presumably does, process an unbounded number of words before connecting them all together.

```
parse ($w_{0,n-1}$)
$C[0](= h =< q, r, t >) \leftarrow < 1, 1, ROOT >$
for $i = 0, n$
    do while ABOVE-THRESHOLD $(h, C, N)$
        remove $h$ from $C$
        for all $x$ such that $p(x \mid t) > 0$
            let $h' =< q', r', t' >$
            where $q' = q * p(x \mid t)$,
                  $r' = \text{LAP}(t', w')$,
                  and $t' = t \circ x$
            if$(x = w)$ then $w' = w_{i+1}$
                  insert $h'$ in $N$
            else $w' = w$
                insert $h'$ in $C$
    empty $C$
    exchange $C$ and $N$
output $t(C[0])$.
```

Figure 1: Roark's Fully-Connected Parsing Algorithm

Here we follow the work of Roark (Roark, 2001) which *is* fully-connected. The basic al-

	Current hypotheses	Prob of next tree element
1	$< 1.4 * 10^{-3}, 5 * 10^{-2}$,(S (NP (NNS Terms)>	p(x=")" | t)=.64
2	$< 7 * 10^{-5}, 5 * 10^{-2}$,(S (S (NP (NNS Terms)>	
3	$< 9 * 10^{-4}, 8 * 10^{-2}$,(S (NP (NNS Terms))>	p(x=VP | t) =.88
4		p(x=S | t)= $2 * 10^{-4}$
5	$< 7 * 10^{-5}, 5 * 10^{-2}$,(S (S (NP (NNS Terms)>	
6	$< 9 * 10^{-4}, 9 * 10^{-2}$,(S (NP (NNS Terms)) (VP>	p(x=AUX | t)= .38
7	$< 7 * 10^{-5}, 5 * 10^{-2}$,(S (S (NP (NNS Terms)>	
8	$< 2 * 10^{-8}, 9 * 10^{-2}$,(S (NP (NNS Terms)) (S>	
9	$< 3 * 10^{-4}, 2 * 10^{-1}$,(S (NP (NNS Terms)) (VP (AUX >	p(x="were" | t)= .21
10	$< 7 * 10^{-5}, 5 * 10^{-2}$,(S (S (NP (NNS Terms)>	
11	$< 2 * 10^{-8}, 9 * 10^{-2}$,(S (NP (NNS Terms)) (S>	
12	$< 7 * 10^{-5}, 3 * 10^{-1}$,(S (NP (NNS Terms)) (VP (AUX were)>	

Figure 2: Parsing the second word of "Terms were not disclosed."

gorithm is given in Figure 1. (Note we have simplified the algorithm in several ways.) The input to the parser is a string of n words $w_{0,n-1}$. We pad the end of the string with an end-of-sentence marker $w_n = \triangleleft$. This has the special property that $p(\triangleleft \mid t) = 1$ for a complete tree t of $w_{0,n-1}$, zero otherwise.

There are two priority queues of hypotheses, C (current), and N (next). A hypothesis h is a three-tuple $< q, r, t >$ where q is the probability assigned to the current tree t. In Figure 1 h always denotes $C[0]$ the top-most element of C. While we call t the "tree", it is a vector representation of a tree. For example, the tree (ROOT) would be a vector of two elements, ROOT and ")", the latter indicating that the constituent labeled root is completed. Thus elements of the vector are the terminals, non-terminals, and ")" — the close parenthesis element. Lastly r is the "look-ahead probability" or LAP. LAP(w,h) is (a crude estimate of) the probability that the next word is w given h. We explain its purpose below.

We go through the words one at a time. At the start of our processing of w_i we have hypotheses on C ordered by $p \cdot q$ — the probability of the hypothesis so far times an estimate q of the probability cost we encounter when trying to now integrate w_i. We remove each h from C and integrate a new tree symbol x. If $x = w_i$ it means that we have successfully added the new word to the tree and this hypothesis goes into the queue for the next word N. Otherwise h does not yet represent an extension to w_i and we put it back on C to compete with the other hypotheses waiting to be extended. The look-ahead probability LAP(h) = q is intended to keep a level playing field. If we put h back onto C it's probability p is lowered by the factor $p(x \mid h)$. On the other hand, if x is the correct symbol, q should go up, so the two should offset and h is still competitive.

We stop processing a word and move onto the next when ABOVE-THRESHOLD returns false. Without going into details, we have adopted exactly the decision criteria and associated parameters used by Roark so that the accuracy numbers presumably reflect the same amount of search. (The more liberal ABOVE-THRESHOLD, the more search, and presumably the more accurate results, everything else being equal.)

Figure 2 shows a few points in the processing of the second word of "Terms were not disclosed." Lines one and two show the current queue at the start of processing. Line one has the ultimately correct partial tree (S (NP (NNS Terms). Note that the NP is not closed off as the parser defers closing constituents until necessary. On the right of line 1 we show the possible next tree pieces that could be added. Here we simply have one, namely a right parenthesis to close off the NP. (In reality there would be many such x's.) The result is that the hypothesis of line 1 is removed from the queue, and a new hypothesis is added back on C as this new hypothesis does not incorporate the second word.

Lines 3 and 5 now show the new state of C. Again we remove the top candidate from C. The right-hand side of lines 3 and 4 show two possible continuations for the h of line 3, start a new VP or a new S. With line 3 removed from the queue, and its two extensions added, we get the new queue state shown in lines 6,7 and 8. Line 6 shows the top-most hypothesis extended by an AUX. This still has not yet incorporated the next word into the parse, so this extension is inserted in the current queue giving us the queue state shown in 9,10,11. Finally line 9 is extended with the word "were." This addition incorporates the current word, and the resulting extension, shown in line 12 is inserted in N, not C, ending this example.

3 Random Forests

The Roark model we emulate requires the estimation of two probability distributions: one for the next tree element (non-terminals,terminals, and ")") in the grammar, and one for the look-ahead probability of the yet-to-be-incorporated next word. In this section we use the first of these for illustration.

We first consider how to construct a single (non-random) decision tree for estimating this distribution. A tree is a fully-connected directed acyclic graph where each node has one input arc (except for the root) and, for reasons we go into later, either zero or two output arcs — the tree is binary. A node is a four-tuple $< d, s, p, q >$, where d is a set of training instances, p, a probability distribution of the correct decisions for all of the examples in d, and q a binary question about the conditioning information for the examples in d. The 0/1 answer to this question causes the decision-tree program to follow the left/right arc out of the node to the children nodes. If q is null, the node is a leaf node. s is a strict subset of the domain of the q for the parent of h.

Decision tree construction starts with the root node n where d consists of the several million situations in the training data where the next tree element needs to be guessed (according to our probability distribution) based upon previous words and the analysis so far. At each iteration one node is selected from a queue of unprocessed nodes. A question q is selected, and based upon its answers two descendents n_1 and n_2 are created with d_1 and d_2 respectively, $d_1 \cup d_2 = d$. These are inserted in the queue of unprocessed nodes and the process repeats. Termination can be handled in multiple ways. We have chosen to simply pick the number of nodes we create. Nodes left on the queue are the leaf nodes of the decision tree. We pick nodes from the heap based upon how much they increased the probability of the data.

Still open is the selection of q at each iteration. First pick a query type qt from a user supplied set. In our case there are 27 types. Examples include the parent of the non-terminal to be created, its predecessor, 2 previous, etc. A complete list is given in Figure 4. Note that the answers to these queries are *not* binary.

Secondly we turn our qt into a binary question by creating two disjoint sets s_1 and s_2 $s_1 \cup s_2 = s$ where s is the domain of qt. If a particular history $h \in d$ is such that $qt(h) = x$ and $x \in s_1$ then h is put in d_1. Similarly for s_2. For example, if qt is the parent relation, and the parent in h is NP, then h goes in d_1 iff NP $\in s_1$. We create the sets s_i by initializing them randomly, and then for each $x \in s$ try moving x to the opposite set s_i. If this results in a higher data probability we keep it in its new s_i, otherwise it reverts to it's original s_i. This is repeated until no switch lowers probability. (Or were the a's are individual words, until no more than two words switch.)

We illustrate with a concrete example. One important fact quickly impressed on the cre-

No.	Q	S	p(“of”)	p(“in”)
0	1			
1	1	NN,IN	0.05	0.03
4	1	NNS,IN	0.09	0.06
12	2	RB,IN	0.17	0.11
16	3	PP,WHPP	0.27	0.18
39	20	NP,NX	0.51	0.16
40	20	S,VP	0.0004	0.19

Figure 3: Some nodes in a decision tree for $p(w_i \mid t)$

ator of parsers is the propensity of prepositional phrases (PP) headed by “of” to attach to noun phrases (NP) rather than verb phrases (VP). Here we illustrate how a decision tree for $p(w_i \mid t)$ captures this. Some of the top nodes in this decision tree are shown in Figure 3. Each line gives a node number, Q — the question asked at that node, examples of answers, and probabilities for “of” and “in”. Questions are specified by the question type numbers given in Figure 4 in the next section. Looking at node 0 we see that the first question type is 1 — parent of the proposed word. The children trees are 1 and 2. We see that prepositions (IN) have been put in node 1. Since this is a binary choice, about half the preterminals are covered in this node. To get a feel for who is sharing this node with prepositions each line gives two examples. For node 1 this includes a lot of very different types, including NN (common singular noun).

Node 1 again asks about the preterminal, leading to node 4. At this point NN has split off, but NNS (common plural noun) is still there. Node 4 again asks about the preterminal, leading to node 12. By this point IN is only grouped with things that are much closer, e.g. RB (adverb).

Also note that at each node we give the probability of both “of” and “to” given the questions and answers leading to that node. We can see that the probability of “of” goes up from 0.05 at node 1 to 0.27 at node 16. The probabilities for “to” go in lockstep. By node 16 we are concentrating on prepositions heading prepositional phrases, but nothing has been asked that would distinguish between these two prepositions. However, at node 16 we ask the question “who is the grandparent” leading to nodes 39 and 40. Node 39 is restricted to the answer “noun phrase” and things that look very much like noun phrases — e.g., NX, a catchall for abnormal noun phrases, while 40 is restricted to PP’s attaching to VP’s and S’s. At this point note how the probability of “of” dramatically increases for node 39, and decreases for 40.

That the tree is binary forces the decision tree to use information about words and nonterminals one bit at a time. In particular, we can now ask for a little information about many different previous words in the sentence.

We go from a single decision tree to a random forest by creating many trees, randomly changing the questions used at every node. First note that in our greedy selection of s_i’s the outcome depends on the initial random assignment of a’s. Secondly, each qt produces its own binary version q. Rather than picking the one that raises the data probability the most, we choose it with probability m. With probability $1 - m$ we repeat this procedure on the list of q’s minus the best. Given a forest of f trees we compute the final probability by taking the average over all the trees:

$$p(x \mid t) = \frac{1}{f} \sum_{j=1,f} p^j(x \mid t)$$

where p^j denotes the probability computed by tree j.

4 Implementation Details

We have twenty seven basic query types as shown in Figure 4. Each entry first gives identification numbers for the query types followed by a description of types. The description is from the point of view of the tree entry x to be added to the tree. So the first line of Figure 4 specifies six query types, the most local of which is the label of the parent of x. For example, if we have the local context “(S (NP (DT the)” and we are assigning a probability to the preterminal after DT, (e.g., NN) then the parent of x is NP. Similarly one of the query types from line two is one-previous, which is DT. Two previous is ϵ, signifying nothing in this position.

1-6 The non-terminal of the parent, grandparent, parent^3, up to parent^6

7-10 The previous non-terminal, 2-previous, up to 4-previous

11-14 The non-terminal just prior to the parent, 2-prior, up to 4 prior

15-16 The non-terminal and terminals of the head of the previous constituent

17-18 Same, but 2 previous

19-20 Same but previous to the parent

21-22 Same but 2 previous to the parent

23-24 The non-terminal and terminal symbols just prior to the start of the current parent constituent

25 The non-terminal prior to the grandparent

26 Depth in tree, binned logarithmically

27 Is a conjoined phrase prior to parent.

Figure 4: Question types

Random forests, at least in obvious implementations, are somewhat time intensive. Thus we have restricted their use to the distribution $p(x \mid t)$. The forest size we picked is 4400 nodes. For the look-ahead probability, LAP, we use a single decision tree with greedy optimal questions and 1600 nodes.

We smooth our random forest probabilities by successively backing off to distributions three earlier in the decision tree. We use linear interpolation so

$$p_l(x \mid t) = \lambda(c_l) * \hat{p}_l(x \mid t) + (1 - \lambda(c_l)) * p_{l-3}(x \mid t)$$

Here p_l is the smoothed distribution for level l of the tree and $\hat{p}_l$ is the maximum likelihood (unsmoothed) distribution. We use Chen smoothing so the linear interpolation parameters λ are functions of the Chen number of the level l node. See (Chen and Goodman, 1996). We could back off to $l - 1$, but this would slow the algorithm, and seemed unnecessary.

Following (Klein and Manning, 2003) we handle unknown and rare words by replacing them with one of about twenty unknown word types. For example, “barricading” would be replaced by UNK-ING, denoting an unknown word ending in “ing.” Any word that occurs less than twenty times in the training corpus is considered rare. The only information that is retained about it is the parts of speech with which it has appeared. Future uses are restricted to these pre-terminals.

Because random forests have so much latitude in picking combinations of words for specific situations we have the impression that it can overfit the training data, although we have not done an explicit study to confirm this. As a mild corrective we only allow verbs appearing 75 times or more, and all other words appearing 250 times or more, to be conditioned upon in question types 16, 18, 20, 22, and 27. Because the inner loop of random-forest training involves moving a conditioning event to the other decedent node to see if this raises training data probability, this also substantially speeds up training time.

Lastly Roark obtained the results we quote here with selective use of left-corner transforms (Demers, 1977; Johnson and Roark, 2000). We also use this technique but the details differ. Roark uses left-corner transforms only for immediately recursive NP's, the most common situation by far. As it was less trouble to do so, we use them for any immediately recursive constituent. However, we are also aware that in some respects left-corner transforms work against the fully-connected tree rule as operationalizing the “understand as you go along” cognitive constraint. For example, the normal sentence initial NP serves as the subject of the sentence. However in Penn-treebank grammar style an initial NP could also be a possessive NP as in (S (NP (NP (DT The) (NN dog) (POS 's)))) Clearly this NP is *not* the subject. Thus using left corner transforms on all NP's allows the parser to conflate differing semantic situations into a single tree. To avoid this we have added the additional restriction that we only allow left-corner treatment when the head words (and thus presumably the meaning) are

	Precision	Recall	F
Collins 2003	88.3	88.1	88.2
Charniak 2000	89.6	89.5	89.6
C/J 2005	91.2	90.9	91.1
Petrov et.al. 2006	90.3	90.0	90.2
Roark 2001	87.1	86.6	86.8
C/R Perceptron	87.0	86.3	86.6
C/R Combined	89.1	88.4	88.8
This paper	**89.8**	**89.0**	**89.4**

Figure 5: Precision/Recall measurements, Penn Treebank Section 23, Sentence length ≤ 100

the same. (Generally head-word rules dictate that the POS is the head of the possessive NP.)

5 Results and Analysis

We trained the parser on the standard sections 2-21 of the Penn Tree-bank, and tested on all sentences of length ≤ 100 of section 23. We used section 24 for development.

Figure 5 shows the performance of our model (last line, in bold) along with the performance of other parsers. The first group of results show the performance of standard parsers now in use. While our performance of 89.4% f-measure needs improvement before it would be worth-while using this parser for routine work, it has moved past the accuracy of the Collins-Bikel (Collins, 2003; Bikel, 2004) parser and is not statistically distinguishable from (Charniak, 2000).

The middle group of results in Figure 5 show a very significant improvement over the original Roark parser, (89.4% vs.86.8%). Although we have not discussed it to this point, (Collins and Roark, 2004) present a perceptron algorithm for use with the Roark architecture. As seen above (C/R Perceptron), this does not give any improvement over the original Roark model. As is invariably the case, when combined the two models perform much better than either by itself (C/R Combined — 88.8%). However we still achieve a 0.6% improvement over that result. Naturally, a new combination using our parser would almost surely register another significant gain.

Conditioning Non-terminals	Conditioning Terminals	F-measure
8	1	86.6
10	2	88.0
13	3	88.3
17	4	88.8
21	6	89.0

Figure 6: Labeled precision-recall results on section 24 of the Penn Tree-bank. All but one sentence of length ≤ 100. (Last one not parsed).

In Figure 6 we show results illustrating how parser performance improves as the probability distributions are conditioned on more diverse information from the partial trees. The first line has results when we condition on only the "closest" eight non-terminal and the previous word. We successively add more distant conditioning events. The last line (89.0% F-measure) corresponds to our complete model but since we are experimenting here on the development set the result is not the same as in Figure 5. (The result is consistent with the parsing community's observation that the test set is slightly easier than the development set — e.g., average sentence length is less.)

One other illustrative result: if we keep all system settings constant and replace the random forest mechanism by a single greedy optimal decision tree for probability computation, performance is reduced to 86.3% f-measure. While this almost certainly overstates the performance improvement due to many random trees (the system parameters could be better adjusted to the one-tree case), it strongly suggests that nothing like our performance could have been obtained without the forests in random forests.

6 Conclusions and Future Work

We have presented a new top-down left-to-right parsing model. Its performance of 89.4% is a 20% error reduction over the previous single-parser performance, and indeed is a small improvement (0.6%) over the best combination-parser result. The code is publically available.[1]

[1] http://bllip.cs.brown.edu/resources.shtml#software

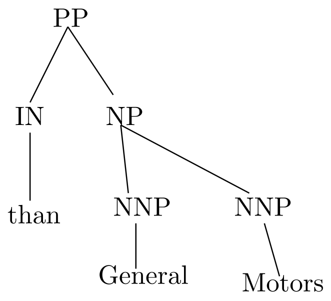


Figure 7: The start of an incorrect analysis for "than General Motors is worth"

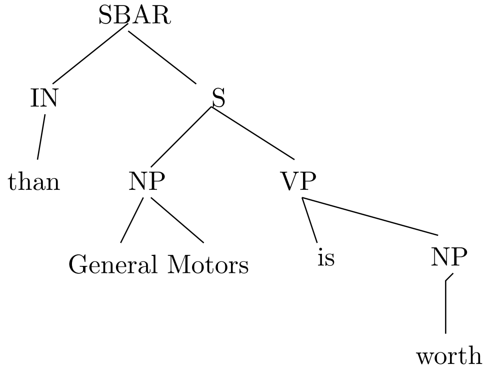


Figure 8: The correct analysis for "than General Motors is worth"

Furthermore, as models of this sort have received comparatively little attention, it seems reasonable to think that significant further improvements may be found.

One particular topic in need of more study is search errors. Consider the following example:

> The government had to spend more than General Motors is worth.

which is difficult for our parser. The problem is integrating the words starting with "than General Motors." Initially the parser believes that this is a prepositional phrase as shown in Figure 7. However, the correct tree-bank parse incorporates a subordinate sentential clause"than General Motors is worth", as in Figure 8. Unfortunately, before it gets to "is" which disambiguates the two alternatives, the subordinate clause version has fallen out of the parser's beam (unless, of course, one sets the beam-width to an unacceptably high level). Furthermore, it does not seem that there is any information available when one starts working on "than" to allow a person to immediately pick the correct continuation. It is also the case that the parsing model gives the correct parse a higher probability if it is available, showing that this is a search error, not a model error.

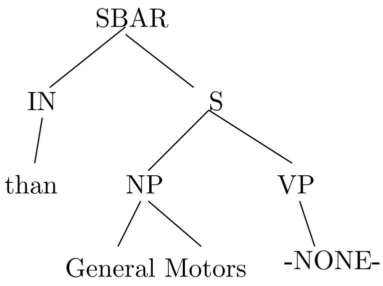


Figure 9: Alternative analysis for "than General Motors"

If there is no information that would allow a person to make the correct decision in time, perhaps people do not need to make this decision. Rather the problem could be in the treebank representation itself. Suppose we reanalyzed "than General Motors" in this context as in Figure 9. Here we would not need to guess anything in advance of the (missing) VP. Furthermore, we can make this change without loosing the great benefit of the treebank for training and testing. The change is local and deterministic. We can tree-transform the training data and then untransform before scoring. It is our impression that a few examples like this would remove a large set of current search errors.

Three other kinds of information are often added as additional annotations to syntactic trees: Penn-Treebank form-function tags, trace elements, and semantic roles. Most research on such annotation takes the parsing process as fixed and is solely concerned with improving the retrieval of the annotation in question. When they have been integrated with parsing, finding the parse and the further annotation jointly has not improved the parse. While it is certainly possible that this would prove to be the same for this new model, the use of random forests to integrate more diverse information sources might help us to reverse this state of affairs.

Finally there is no reason why we even need to stop our collection of features at sentence boundaries — information from previous sentences is there for our perusal. There are many known intra-sentence correlations, for example "sentences" that are actually fragments are much more common if the previous sentence is a question. The tense of sequential sentences main verbs are correlated. Main clause subjects are more likely to be co-referent. Certainly the "understanding" humans pick up helps them assign structure to subsequent phrases. How much, if any, of this meaning we can glean given our current (lack-of) understanding of semantics and pragmatics is an interesting question.

7 Acknowledgements

Thanks to members of BLLIP and Mark Johnson who read earlier drafts of this paper. Also thanks to Brian Roark and Mark Johnson for pointers to previous work.

References

Stephen Abney and Mark Johnson. 1991. Memory requirements and local ambiguities of parsing strategies. *Journal of Psycholinguistic Research*, 20(3):233–250.

Y. Amit and D. Geman. 1997. Shape quantization and recognition with randomized trees. *Neural Computation*, 9:1545–1588.

Dan Bikel. 2004. *On the Parameter Space of Lexicalized Statistical Parsing Models*. Ph.D. thesis, University of Pennsylvania.

Leo Breiman. 2004. Random forests. *Machine Learning*, 45(1):5–32.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 173–180, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *The Proceedings of the North American Chapter of the Association for Computational Linguistics*, pages 132–139.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In Arivind Joshi and Martha Palmer, editors, *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 310–318, San Francisco. Morgan Kaufmann Publishers.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 111–118, Barcelona, Spain, July.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–638.

A. Demers. 1977. Generalized left-corner parsing. In *Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, 1977 ACM SIGACT/SIGPLAN*, pages 170–182.

Jay Earley. 1970. An efficient contex-free parsing algorithm. *Communications of the ACM*, 6(8):451–445.

Suzan L. Graham, Michael Harrison, and Walter L. Ruzzo. 1980. An improved context-free recognizer. *ACM Transations on Programming Languages and Systems*, 2(3):415–462.

James Henderson. 2003. Inducing history representations for broad coverage statistical parsing. In *Proceedings of HLT-NAACL 2003*.

Mark Johnson and Brian Roark. 2000. Compact non-left-recursive grammars using the selective left-corner transform and factoring. In *Proceedings of COLING-2000*.

Dan Klein and Christopher Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*.

Michell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia, July. Association for Computational Linguistics.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

Roger van Gompel and Martin J. Pickering. 2007. Syntactic parsing. In G. Gatskil, editor, *The Oxford handbook of psycholinguistics*. Oxford University Press.

Peng Xu and Frederick Jelinek. 2004. Random forests in language modeling. In *Proceedings of the 2004 Empirical Methods in Natural Language Processing Conference*. The Association for Computational Linguistics.

Improved Fully Unsupervised Parsing with Zoomed Learning

Roi Reichart
ICNC
The Hebrew University
roiri@cs.huji.ac.il

Ari Rappoport
Institute of computer science
The Hebrew University
arir@cs.huji.ac.il

Abstract

We introduce a novel training algorithm for unsupervised grammar induction, called *Zoomed Learning*. Given a training set T and a test set S, the goal of our algorithm is to identify subset pairs T_i, S_i of T and S such that when the unsupervised parser is trained on a training subset T_i its results on its paired test subset S_i are better than when it is trained on the entire training set T. A successful application of zoomed learning improves overall performance on the full test set S.
We study our algorithm's effect on the leading algorithm for the task of fully unsupervised parsing (Seginer, 2007) in three different English domains, WSJ, BROWN and GENIA, and show that it improves the parser F-score by up to 4.47%.

1 Introduction

Grammar induction is the task of learning grammatical structure from plain text without human supervision. The task is of great importance both for the understanding of human language acquisition and since its output can be used by NLP applications, avoiding the costly and error prone creation of manually annotated corpora. Many recent works have addressed the task (e.g. (Klein and Manning, 2004; Seginer, 2007; Cohen and Smith, 2009; Headden et al., 2009)) and its importance has increased due to the recent availability of huge corpora.

A basic challenge to this research direction is how to utilize training data in the best possible way. Klein and Manning (2004) report results for their dependency model with valence (DMV) for unsupervised dependency parsing when it is trained and tested on the same corpus (both when sentence length restriction is imposed, such as for WSJ10, and when it is not, such as for the entire WSJ). Today's best unsupervised dependency parsers, which are rooted in this model, train on short sentences only: both Headen et al., (2009) and Cohen and Smith (2009) train on WSJ10 even when the test set includes longer sentences.

Recently, Spitkovsky et al., (2010) demonstrated that training the DMV model on sentences of up to 15 words length yields better results on the entire section 23 of WSJ (with no sentence length restriction) than training with the entire WSJ corpus.

In contrast to these dependency models, the Seginer constituency parser achieves its best performance when trained on the entire WSJ corpus either if sentence length restriction is imposed on the test corpus or not. The sentence length restriction training protocol of (Spitkovsky et al., 2010), harms this parser. When the parser is trained with the entire WSJ corpus its F-score performance on the WSJ10, WSJ20 and the entire WSJ corpora are 76, 64.8 and 56.7 respectively. When training is done with WSJ10 (WSJ20) performance degrades to 60 (72.2), 37.4 (61.9) and 29.7 (48) respectively.

In this paper we introduce the *Zoomed Learning* (ZL) technique for unsupervised parser training: given a training set T and a test set S, it identifies subset pairs T_i, S_i of T and S such that when the unsupervised parser is trained on a training subset T_i its results on its paired test subset S_i are better than when it is trained on the entire training set T. A

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 684–693,
MIT, Massachusetts, USA, 9-11 October 2010.

successful application of zoomed learning improves performance on the full test set S.

We describe ZL algorithms of increasing sophistication. In the simplest algorithm the subsets are randomly selected while in the more sophisticated versions subset selection is done using a fully unsupervised measure of constituency parse tree quality.

We apply ZL to the Seginer parser, the best algorithm for fully unsupervised constituency parsing. The input is a plain text corpus without any annotation, not even POS tagging[1], and the output is an unlabeled bracketing for each sentence.

We experiment in three different English domains: WSJ (economic newspaper), GENIA (biological articles) and BROWN (heterogeneous domains), and show that ZL improves the parser F-score by as much as 4.47%.

2 Related Work

Unsupervised parsing has attracted researchers for over a quarter of a century (see (Clark, 2001; Klein, 2005) for reviews). In recent years efforts have been made to evaluate the algorithms on manually annotated corpora such as the WSJ PennTreebank. Recent works on unlabeled bracketing or dependencies induction include (Klein and Manning, 2002; Klein and Manning, 2004; Dennis, 2005; Bod, 2006a; Bod, 2006b; Bod, 2007; Smith and Eisner, 2006; Seginer, 2007; Cohen et al., 2008; Cohen and Smith, 2009; Headden et al., 2009). Most of the works above use POS tag sequences, created either manually or by a supervised algorithm, as input. The only exception is Seginer's parser, which induces bracketing from plain text.

Our confidence-based ZL algorithms use the PUPA unsupervised parsing quality score (Reichart and Rappoport, 2009b). As far as we know, PUPA is the only unsupervised quality assessment algorithm for syntactic parsers that has been proposed. Combining PUPA with Seginer's parser thus preserves the fully unsupervised nature of the task.

Quality assessment of a learning algorithm's output has been addressed for supervised algorithms (see (Caruana and Niculescu-Mizil, 2006) for a survey) and specifically for supervised syntactic parsing (Yates et al., 2006; Reichart and Rappoport, 2007; Ravi et al., 2008; Kawahara and Uchimoto, 2008). All these algorithms are based on manually annotated data and thus do not preserve the unsupervised nature of the task addressed in this paper.

We experiment with the Seginer parser for two reasons. First, this is the best algorithm for the task of fully unsupervised parsing which motivates us to improve its performance. Second, this is the only publicly available unsupervised parser that induces constituency trees. The PUPA score we use in our confidence-based algorithms is applicable for constituency trees only. When additional constituency parsers will be made available, we will test ZL with them as well. Interestingly, the results reported for other constituency models (the CCM model (Klein and Manning, 2002) and the U-DOP model (Bod, 2006a; Bod, 2006b)) are reported when the parser is trained on its test corpus even if the sentences is that corpus are of bounded length (e.g. WSJ10). This raises the question if using more training data (e.g. the entire WSJ) wisely can enhance these models.

Recently, Spitkovsky et al., (2010) proposed three approaches for improvement of unsupervised grammar induction by considering the complexity of the training data. The approaches have been applied to the DMV unsupervised dependency parser (Klein and Manning, 2004) and improved its performance. One of these approaches is to train the model with sentences whose length is up to 15 words. As noted above, such a training protocol fails to improve the performance of the Seginer parser.

The other approaches in that paper, bootstrapping via iterated learning of increasingly longer sentences and a combination of the bootstrapping and the short sentences approaches, are not directly applicable to the Seginer parser since its training method cannot be trivially bootstrapped with parses created in former steps (Seginer, 2007).

Related machine learning methods. ZL is related to ensemble methods. Both ZL and such methods produce multiple learners, each of them trained on a different subset of the training data, and decide which learner to use for a particular test instance. Bagging (Breiman, 1996) and boosting (Freund and Schapire, 1996), where the experts utilize the same

[1] For clarity of exposition, we still refer to this corpus as our *training corpus*. In the algorithms presented in this paper, the test set is included in the training set which is a common practice in unsupervised parsing.

learning algorithm and differ in the sample of the training data they use for its training, were applied to supervised parsing (Henderson and Brill, 2000; Becker and Osborne, 2005). In Section 3 we discuss the connection of ZL to boosting.

Owing to the fact that ZL produces different learners, it is natural to use it in conjunction with an ensemble method, which is what we do in this paper with our EZL model (Section 3).

ZL is also related to active learning (AL) (Cohn and Ladner, 1994). AL also uses training subset selection, with the goal of obtaining a faster learning curve for an algorithm. AL is done in supervised settings, usually in order to minimize human annotation costs. AL algorithms providing faster learning than random subset selection for parsing have been proposed (Reichart and Rappoport, 2009a; Hwa, 2004). However, we are not aware of AL applications in which the *overall* performance on the test set has been improved. In addition, our application here is to an unsupervised problem.

Algorithms that utilize unsupervised clustering for class decomposition in order to improve classifiers' performance (e.g. (Vilalta and Rish, 2003)) are related to ZL. In such methods, examples that belong to the same class are clustered, and the induced clusters are considered as separate classes. These methods, however, have been applied only to supervised classification in contrast to our work that addresses unsupervised structured learning. Moreover, after class decomposition a classifier is trained with the entire training data while the subsets identified by a ZL algorithm are parsed by a parser trained only with the sentences they contain.

3 Zoomed Learning Algorithms

Zoomed Learning proposes that performance on a particular test instance might improve if training is done on a proper *subset* of the training set. The ZL view is clearly applicable when the training data is comprised of subsets originating from different sources having different natures. If the test data is also similarly composed, performance on any particular test instance might improve if training is done on a training subset coming from the same source. However, even when the training and test data are from the same source, a ZL algorithm may capture fine differences between subsets.

The ZL idea is therefore related to the notions of *in-domain* and *out-of-domain* (domain adaptation). In the former, the training and test data are assumed to originate from the same domain. In the latter, the test data comes from a different domain, and therefore has different statistics from the training data. Indeed, the performance of NLP algorithms in domain adaptation scenarios is markedly lower than in in-domain ones (McClosky et al., 2006).

ZL takes this observation to the extreme, assuming that a similar situation might exist *even in in-domain scenarios.* After all, a 'domain' is only a coarse qualification of the nature of a data set. In NLP, a domain is usually specified as the genre of the text involved (e.g., 'newspapers'). However, there are additional axes that might influence the statistics obtained from training data, e.g., the syntactic nature of sentences.

This section presents our ZL algorithms. We start with the simplest possible ZL algorithm where the subsets are randomly selected. We then describe ZL algorithms based on quality-based parse selection. We first detail a basic version and then an extended version consisting of another level of parse selection. Finally, we briefly discuss the PUPA quality measure that we use to evaluate the quality of a parse tree.

In all versions of the algorithm the input consists of a set T of N training sentences, a set $S \subseteq T$ of test sentences, and an integer number $N_H \leq N$.

Zoomed Learning with Random Selection (RZL). The simplest ZL algorithm randomly assigns each of the training sentences to one of n sets ($n = 2$ in this paper). More explicitly, the set number is drawn from a uniform distribution on $\{1, 2, \ldots n\}$. Each set is then parsed by a parser that is trained only with the sentences contained in that set.

The intuition behind this algorithm is that different sets of sentences are likely to manifest different syntactic patterns. Consequently, the best way to learn the syntactic patterns of any given set of sentences might be to train the parser on the sentences contained in the set.

While simple, in Section 5 it is shown to improve the performance of the Seginer parser.

The Basic Quality-Based Algorithm (BZL). The idea of the basic ZL algorithm is that sentences for which the parser provides low quality parses manifest different syntactic patterns than the sentences for which the parser provides high quality parses. The main challenge is therefore to estimate the quality of the produced parses without supervision.

The algorithm has three stages. In the first, we create the fully-trained model by training the parser using all of the N sentences of T. We then parse these N sentences using the fully-trained model.

In the second, we compute a parse confidence score for each of the N sentences, based on the N parses produced in the first stage. We divide the training sentences to two subsets: a high quality subset H consisting of the top scored N_H sentences, and a lower quality subset L consisting of the other $N_L = N - N_H$ sentences.

As is common practice for this problem (Klein and Manning, 2004; Seginer, 2007), the test set is contained in the training set. This methodology is a valid one because the training set is unannotated. Our test set is thus naturally divided into two subsets, a high quality subset HT consisting of the test set sentences contained in H and a lower quality subset LT consisting of the test set sentences contained in L.

In the third stage, each of the test subsets is parsed by a model trained only on its corresponding training subset. This stage is motivated by our assumption that the high and low quality subsets manifest dissimilar syntactic patterns, and consequently the statistics of the parser's parameters suitable for one subset differ from those suitable for another.

We compute the confidence score in the second stage using the unsupervised PUPA algorithm (Reichart and Rappoport, 2009b). POS tags for it are induced using the fully unsupervised algorithm of Clark (2003). The parser we experiment with is the incremental parser of Seginer (2007), whose input consists of raw sentences and does not include any kind of supervised POS tags (created either manually or by a supervised algorithm). Consequently, our algorithm is fully unsupervised. The only parameter it has is N_H but ZL improves parser performance for most N_H values.

BZL is related to boosting. In boosting after training one member of the ensemble, examples are reweighted such that examples that are classified correctly are down-weighted. BZL does something similar: it uses PUPA to estimate which sentences are given high quality parse trees, and down-weights examples with high (low) PUPA score to 0 when training the L-trained (H-trained) model. However, in boosting the entire test set is annotated by the same learning model, while ZL parses each test subset with a model trained on its corresponding training subset.

The Extended Quality-Based Algorithm (EZL). The basic algorithm produces an ensemble of two parsing experts: the one trained on H and the one trained on L. It uses the ensemble to parse the test set by applying the H-trained expert to HT and the L-trained expert to LT. Naturally, there are other ways to utilize the ensemble to parse the test set. In addition, even if parse trees generated by the experts are better with high probability than those of the fully trained parser, they are not guaranteed to be so. The fully trained parser is therefore also a valuable member in the ensemble. Consequently, we introduce an extended zoomed learning algorithm (EZL).

The extended version is implemented as a final fourth stage of the previously described basic algorithm. In this stage, the two test subsets are parsed by the fully trained parsing model, in addition to being parsed by the zooming parsing models. We now have two parses for each test sentence s: $P_Z(s)$, the parse created by a parser trained with the sentences contained in its corresponding training subset, and $P_F(s)$, created by the fully trained parser.

For each of the two parses of each test sentence, a confidence score is computed by PUPA. As will be reviewed below, PUPA uses a *set* of parsed sentences to compute the statistics on which its scores are based. Therefore, there are two sources for a difference between the scores of the two parse trees of a given test sentence: the difference between the trees themselves, and the difference between the parses of the other sentences in the set.

The PUPA score for $P_Z(s)$ is computed using the parses created for the sentences contained in the test subset of s by a parser trained with the corresponding training subset. The PUPA score for $P_F(s)$ is computed using the parses created for the entire test set by the fully trained parser.

The algorithm now outputs a final parse by selecting for each sentence the parse tree having the higher PUPA score.

The PUPA Confidence Score. In the second and fourth stages of the confidence-based algorithms, an unsupervised confidence score is computed for each of the induced parse trees. The confidence score algorithm we use is the POS-based Unsupervised Parse Assessment (PUPA) algorithm (Reichart and Rappoport, 2009b). We provide here a brief description of this algorithm.

The input to PUPA is a set I of parsed sentences, and its output consists of a confidence score in $[0, 1]$ assigned to each sentence in I.

The PUPA algorithm collects statistics of the syntactic structures (parse tree constituents) contained in the set I of parsed sentences. The constituent representation is based on the POS tags of the words in the yield of the constituent and of the words in the yields of neighboring constituents. We follow Reichart and Rappoport (2009b) and induce the POS tags using the fully unsupervised POS induction algorithm of Clark (2003).

The algorithm then goes over each individual tree in the set I and scores it according to the collected statistics The PUPA algorithm is guided by the idea that syntactic structures that are frequently created by the parser are more likely to be correct than structures the parser produces less frequently. Therefore, constituents that are more frequent in the set I receive higher scores after proper regularization is applied to prevent potential biases. The tree score is a combination of the scores of its constituents.

Full details of the PUPA algorithm are given in (Reichart and Rappoport, 2009b). The resulting score was shown to be strongly correlated with the extrinsic quality of the parse tree, defined to be its F-score similarity to the manually created (gold standard) parse tree of the sentence.

4 Experimental Setup

We experimented with three English corpora: the WSJ Penn Treebank (Marcus et al., 1993) consisting of economic newspaper texts, the BROWN corpus (Francis and Kucera, 1979) consisting of texts of various English genres (e.g. fiction, humor, romance, mystery and adventure) and the GENIA corpus (Kim et al., 2003) consisting of abstracts of scientific articles from the biological domain. All corpora were stripped of all annotation (bracketing and POS tags).

For all corpora we report the parser performance on the entire corpus (WSJ: 49206 sentences, BROWN: 24243 sentences, GENIA: 4661 sentences). For WSJ we also provide an analysis of the performance of the parser when applied to sentences of bounded length. These sub-corpora are WSJ10 (7422 sentences), WSJ20 (25522 sentences) and WSJ40 (47513 sentences) where WSJY denotes the subset of WSJ containing sentences of length at most Y (excluding punctuation).

Seginer's parser achieves its best reported results when trained on the full WSJ corpus. Consequently, for all corpora, we compare the performance of the parser when trained with the ZL algorithms to its performance when trained with the full corpus.

The POS tags required as input by the PUPA algorithm are induced by the fully unsupervised POS induction algorithm of Clark (2003)[2]. Reichart and Rappoport (2009b) demonstrated an unsupervised technique for the estimation of the number of induced POS tags with which the correlation between PUPA's score and the parse F-score is maximized. When exploring an experimental setup identical to our WSJ setup, they set the number of induced tags to be 5. We therefore induced 5 POS tags for each corpus, using all its sentences as input for Clark's algorithm. Our implementation of the PUPA algorithm will be made available on line.

For each corpus we performed K experiments with each of the three ZL algorithms, where K equals to the number of sentences in the corpus divided by 1000 (rounded upwards). In each experiment the size of the high quality H and lower quality L training subsets is different. H consists of the N_H top ranked sentences according to PUPA (or N_H randomly selected sentences for RZL), with N_H changing from 1000 upwards in steps of 1000. L consists of the rest of the sentences in the training corpus (WSJ). The results reported for RZL are averaged over 10 runs.

We report the parser performance on the test corpus for each training protocol. Following the unsupervised parsing literature multiple brackets and brackets covering a single word are not counted, but the sentence level bracket is. We exclude punctua-

[2] www.cs.rhul.ac.uk/home/alexc/RHUL/Downloads.html

	WSJ10, F(Full) = 76			WSJ20, F(Full) = 64.82			WSJ40, F(Full) = 57.54			WSJ, F(Full) = 56.7		
N_H	25%	50%	75%	25%	50%	75%	25%	50%	75%	25%	50%	75%
EZL	**76.38** **+0.38**	**76.80** **+0.80**	**76.14** **+0.14**	**65.75** **+0.93**	**66.14** **+1.30**	**65.66** **+0.82**	**58.32** **+0.78**	**58.75** **+1.21**	**58.56** **+1.02**	**57.47** **+0.77**	**57.90** **+1.20**	**57.73** **+ 1.13**
BZL	75.07 -0.93	75.78 -0.22	75.02 -0.98	65.08 +0.26	65.74 +0.92	64.79 -0.03	58.13 +0.59	58.70 +1.16	58.21 +0.67	57.30 +0.60	57.88 +1.18	57.66 +1.06
RZL	75.41 -0.59	75.00 -1.00	75.32 -0.68	64.43 -0.39	64.66 -0.16	65.32 +0.50	57.27 -0.27	57.63 +0.09	58.39 +0.85	56.44 -0.26	56.84 +0.14	57.59 +0.89

	WSJ10			WSJ20			WSJ40			WSJ		
$\|LT\|$	10%	20%	30%	10%	20%	30%	10%	20%	30%	10%	20%	30%
EZL	**1.32**	**0.95**	**0.61**	**2.98**	**3.13**	**1.76**	**2.60**	**2.80**	2.62	**2.44**	2.40	2.50
BZL	0.37	0.80	0.53	2.38	3.12	1.23	2.34	3.20	**3.35**	2.28	**2.50**	**3.23**
RZL	-2.10	-1.88	-1.20	-0.91	-0.50	0.72	0.30	0.35	1.50	0.34	0.50	1.60

Table 1: Performance of the EZL, BZL and RZL algorithms in the WSJ experiments (results for BROWN and GENIA are shown in Table 2). Results are presented for four test corpora WSJ10, WSJ20, WSJ40 and the entire WSJ. **Top table:** Results for various values of N_H (the number of sentences in the high quality training subset). Evaluation is performed for all sentences in the test corpora. For each algorithm, the top line is its F-score performance and the bottom line is the difference from the F-score of the fully-trained Seginer parser (denoted by F(Full)). The EZL algorithm is superior. **Bottom table:** Results for various lower quality test subsets. Presented are the differences from the F-score of the fully-trained Seginer parser. The test subsets selected by different algorithms for a specific N_H value are not necessarily identical and for the sub-corpora they are not necessarily of identical size. Reported are the improvements for the LT's of smallest size which is over 10%, 20%, and 30% of the test corpus (the top table reports results for the entire test set, which is why we can report F-scores there). The LT set size is denoted with $|LT|$.

tion and null elements as in (Klein, 2005). To evaluate the quality of a parse tree with respect to its gold standard, the unlabeled parsing F-score is used.

5 Results

Entire Corpus Results. We start by discussing the effect of ZL on the performance of the Seginer parser when no length restriction is imposed on the test corpus sentences (WSJ, BROWN and GENIA).

Table 1 (top, right section, for WSJ), Figure 1 (top line, right graph, for WSJ), and Table 2 (the left section of each table, top table for BROWN and bottom table for GENIA) present the difference between the F-score performance of the Seginer parser when trained with the ZL algorithms and the parser's performance when trained with the entire corpus.

For all test corpora and sizes of the high quality training subset (N_H), zoomed learning improves the parser performance. ZL improves the parser performance by 1.13% (WSJ), 1.46% (BROWN, the number does not appear in the table) and 4.47% (GENIA).

For WSJ, the most substantial improvement is provided by EZL, while for BROWN and GENIA the best results for some N_H values are achieved by BZL and for others by EZL (and for GENIA with small N_H values even by RZL).

Note, that for all three corpora zoomed learning with random selection (RZL) improves the parser performance on the entire test corpus, although to a lesser extent than confidence-based ZL. This is true for almost all N_H values, including those that do not appear in the tables. See Figure 1 (top line, rightmost graph) for WSJ.

We follow the unsupervised parsing literature and provide performance analysis for WSJ sentences of bounded length (WSJ10, WSJ20 and WSJ40). To prevent clutter, for BROWN and GENIA we report only entire corpus results.

Table 1 (top, left three sections) and Figure 1 (top line, three leftmost graphs) present results for WSJ10, WSJ20 and WSJ40.

The result patterns for the sub-corpora are similar to those reported for the entire WSJ corpus. EZL and BZL both improve over the fully-trained parser, and

BROWN	ENTIRE CORPUS (F = 57.19)				LT				HT			
N_H	10%	30%	50%	70%	10%	30%	50%	70%	10%	30%	50%	70%
EZL	0.55	0.69	**0.64**	**0.66**	0.65	0.82	**1.15**	**1.31**	-1.44	-0.03	0.04	**0.30**
BZL	**1.11**	**0.80**	0.02	-0.10	**1.42**	**1.20**	0.76	0.51	-4.80	-1.30	-0.79	-0.37
RZL	0.257	0.755	0.49	0.24	0.23	0.75	0.60	0.53	**0.44**	**0.76**	**0.42**	0.12

GENIA	ENTIRE CORPUS (F = 42.71)				LT				HT			
N_H	10%	30%	50%	70%	10%	30%	50%	70%	10%	30%	50%	70%
EZL	0.01	0.83	1.10	1.66	-0.01	0.76	0.80	3.37	0.34	1.00	1.40	1.55
BZL	-0.46	1.40	**2.74**	**4.47**	-0.54	0.40	0.96	**4.09**	0.42	**4.29**	**4.60**	**5.49**
RZL	**0.61**	**1.70**	2.09	1.99	**0.28**	**2.08**	**3.30**	3.86	**3.04**	3.22	2.80	1.85

Table 2: Results for the BROWN (top table) and GENIA (bottom table) corpora. Results are presented for the entire corpus (left column section), the low quality test subset (middle column section, LT) and the high quality test subset (right column section, HT) of each corpus, as a function of the high quality training set size (N_H). Since the tables present entire corpus results, the training and test subsets are identical.

the improvement of the former is more substantial.

Baselines. A key principle of ZL is the selection of subsets that are better parsed by a parser trained only with the sentences they contain than with a parser trained with the entire training corpus. To verify the importance of this principle we considered two alternative training protocols.

In the first, the entire test corpus is parsed with a parser that was trained with a subset of randomly selected sentences from the training set. We run this protocol for all three corpora (and for the WSJ sub-corpora) with various training set sizes and obtained substantial degradation in the parser performance. The performance monotonically increases with the training set size and reached its maximum when the entire corpus is used. We conclude that using less training material harms the parser performance if a test subset is not carefully selected.

The second protocol is the ‘less is more protocol of Spitkovsky et al., (2010) in which we parsed each test corpus using a parser that was trained with all training sentences of a bounded length. Unlike in their paper, in which this protocol improves the perofrmance of the DMV unsupervised dependency parser (Klein and Manning, 2004), for the Seginer parser the protocol harms the results. When parsing the entire WSJ with a WSJ10-trained parser or with a WSJ20-trained parser, the F-score results are 59.99% and 72.22% compared to 76.00% of the fully-trained parser. For GENIA the numbers are 15.61 and 35.87 compared to 42.71 and for BROWN they are 36.05 and 50.02 compared to 57.19 [3].

It is also interesting that sentence length is generally not a good subset selection criterion for ZL. When parsing WSJ10 with a WSJ10-trained parser, F-score results are 59.29 while the F-score of the fully-trained parser on this corpus is 76.00. The same phenomenon is observed with WSJ20 (F-score of 61.90 with WSJ20 training and of 64.82 with the entire WSJ training), and for the BROWN corpus (65.01 vs. 69.43 for BROWN10 and 61.90 vs 62.92 for BROWN20). For GENIA, however, while parsing GENIA10 with a GENIA10-trained parser harms the performance (45.28 vs. 60.23), parsing GENIA20 with a GENIA20-trained parser enhances the performance (53.23 vs. 50.00).

These results emphasize the power of random selection for ZL as random selection does provide a good selection criterion.

LT vs. HT. In what follows we analyze the ZL algorithms aiming to characterize their strengths and weaknesses.

Table 1 (bottom), the middle and right sections of Table 2 (both tables) and Figure 1 (second and third lines) present the performance of the ZL algorithms on the lower quality and higher quality test subsets (LT and HT). The results patterns for WSJ and BROWN are different than those of GENIA.

For WSJ (and its sub-corpora) and BROWN,

[3] We repeated this protocol multiple times for each corpus, training the parser with sentences of length 5 to 45 in steps of 5. In all cases we observed performance degradation compared to the fully-trained parser.

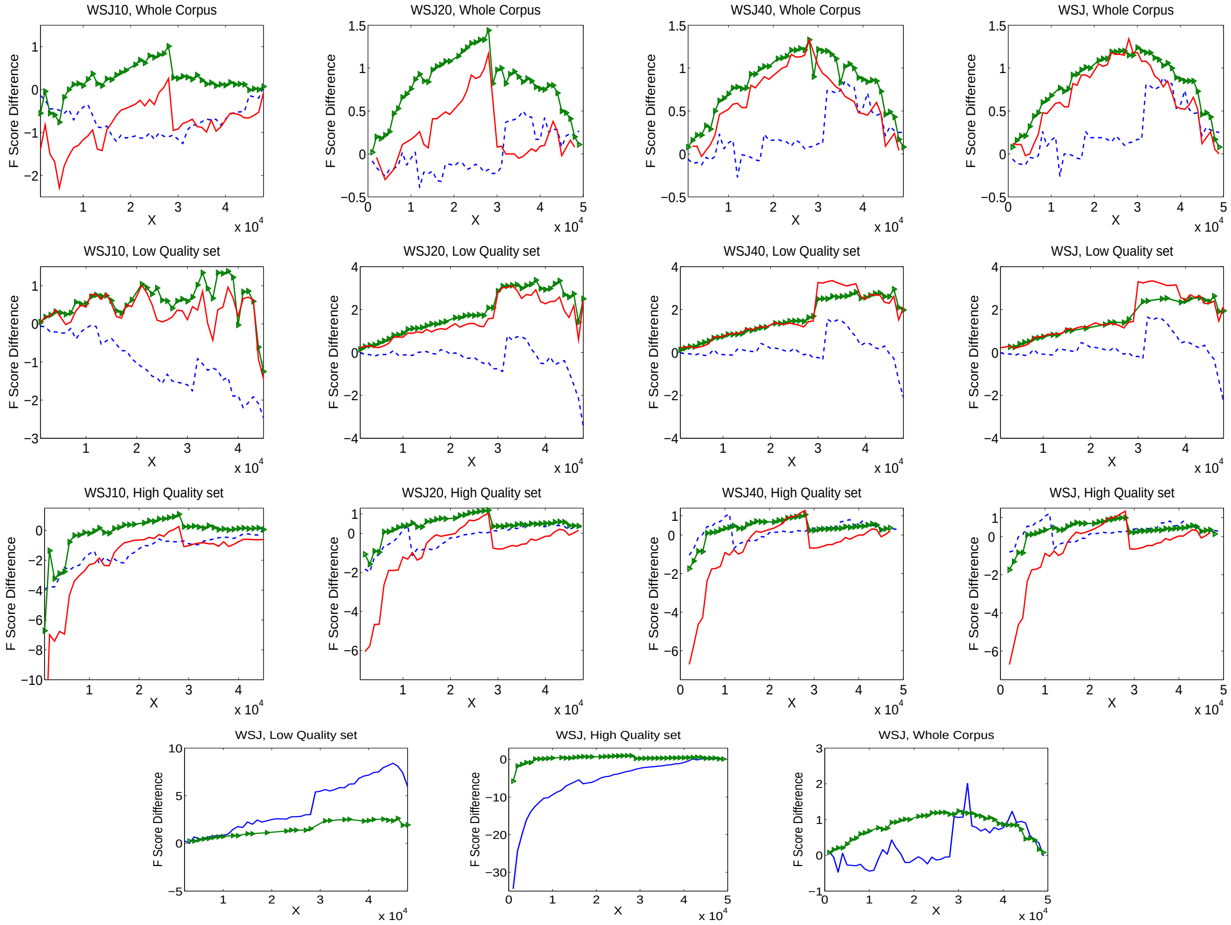


Figure 1: WSJ results. **Top Three Lines:** Difference in F-score performance of the Seginer parser between training with ZL and training with the entire WSJ corpus. Results are presented for the entire corpus (top line), the lower quality test subset (LT, middle line) and the higher quality test subset (HT, bottom line) as a function of the size of the high quality training subset $X = N_H$, measured in sentences. The curve with triangles is for the extended zoomed learning algorithm (EZL), the solid curve is for the basic zoomed learning algorithm (BZL) and the dashed curve is for zoomed learning with random selection (RZL). **Bottom line:** Comparison between the performance of the Seginer parser with the EZL algorithm (curves with triangles) and when subset selection is performed using the oracle F-score of the trees (solid curves). F-score differences from the performance of the fully trained parser are presented for the WSJ test corpus as a function of N_H, the high quality training subset size. Oracle selection is superior for the lower quality subset but inferior for the high quality subset.

confidence-based ZL (BZL and EZL) provides a substantial improvement for LT. For WSJ, F-score improvement is up to 1.32% (WSJ10), 3.13% (WSJ20), 3.35% (WSJ40) and 3.23% (the entire WSJ). For BROWN the improvement is up to 1.42%.

For HT, confidence-based ZL is less effective when these corpora are considered. As indicated in the third line of Figure 1, for WSJ and its subcorpora, EZL leads to a small improvement on HT, while BZL generally leads to a performance degradation on this test subset. For BROWN (the right section of Table 2 (top)), confidence-based ZL generally leads to a performance degradation on HT.

For GENIA, EZL and BZL improve the parser performance on both LT and HT for most N_H values. Understanding this difference is a subject for future research. Our initial hypothesis is that due to the relative small size of the GENIA corpus (4661 sentences compared 24243 and 49206 sentences of WSJ and BROWN respectively), there is more room for improvement in the parser performance on this corpus, and consequently ZL improves on both sets.

Oracle Analysis. Confidence-based ZL is based on the idea that sentences for which the fully-trained parser provides parses of similar quality manifest similar syntactic patterns. Consequently, the parser performance on a set of such sentences can be improved if it is trained only with the sentences contained in the set. An oracle experiment, where selection is based on the F-score computed using the gold standard tree instead of on the PUPA score, can shed light on the validity of this idea.

Figure 1 (bottom line) compares the performance of EZL with that of the oracle-based zoomed learning algorithm when the test corpus is the entire WSJ. For the low quality test subset, oracle selection is dramatically better than confidence-based selection. For the high quality test subset the opposite pattern holds, that is, EZL is superior. These differences lead to the entire corpus pattern where EZL is superior for most N_H values.

Oracle-based and confidence-based zoomed learning demonstrate the same trend: they improve over the baseline for LT much more than for HT. For HT, oracle-based ZL even harms results and so does BZL, which does not benefit from the averaging effect of EZL. The magnitude of the effect of oracle-based zoomed learning is much stronger. These results support our idea that training the parser on a set selected by a well-designed confidence test leads to improvement of the parser performance for the selected sentences when the fully-trained parser produces parses of mediocre quality for them.

Integration of the experimental results for zoomed learning with the three selection methods: random, confidence-based and oracle-based leads to an important conclusion that should guide future research. The more accurate the confidence score used by the zoomed learning algorithm, the more substantial is the performance improvement for the low quality test subset, at the cost of more substantial degradation in the performance on the high quality subset (but recall the different GENIA pattern which should be further explored).

EZL **Variants.** For confidence-based ZL we explored two methods for utilizing the ensemble members for generating a final parse tree for each of the test sentences. In BZL, the L-trained parser and the H-trained parser generate parse trees for LT and HT sentences respectively. In EZL, for each sentence the final parse is selected between the parse created by a parser trained with the sentences contained in its corresponding training subset, and the parse created by the fully trained parser.

There are other ways to use the ensemble members. While for all corpora it is beneficial to use the L-trained parser for the low quality test subset (LT), the results for WSJ and BROWN imply that it might be better to use the fully-trained parser or the EZL algorithm to parse the high quality test subset (HT). We have experimented with these methods and got only a minor improvement over the results reported here (improvement is more substantial for BROWN than for WSJ but does not exceed 0.5% for both). This can also be inferred from the relative minor performance degradation of BZL and EZL on HT.

We also explored a ZL scenario in which the entire test set is parsed either by the H-trained parser or by the L-trained parser. These protocols result in substantial degradation in parser performance (compared to the fully-trained parser) since the performance of the H-trained parser on LT and the performance of the L-trained parser on HT are poor.

6 Conclusions

We introduced zoomed learning – a training algorithm for unsupervised parsers. We applied three variants of ZL to the best fully unsupervised parsing algorithm (Seginer, 2007) and show an improvement of up to 4.47% in three English domains: WSJ, BROWN and GENIA.

Future research should focus on the development of more accurate estimators of parser output quality, and experimentation with different corpora, languages and parsers.

Developing a quality assessment algorithm for dependency trees will allow us to apply confidence-based ZL to unsupervised dependency parsing. Particularly, it will enable us to explore the combination of the methods proposed in (Spitkovsky et al., 2010) with ZL for the DMV model and to integrate the PUPA score into their bootstrapping algorithm.

Another direction is to apply ZL to other NLP tasks and ML areas, supervised and unsupervised.

References

Markus Becker and Miles Osborne, 2005. A two-stage method for active learning of statistical grammars. *IJCAI '05*.

Rens Bod, 2006a. An all-subtrees approach to unsupervised parsing. *ACL-COLING '06*.

Rens Bod, 2006b. Unsupervised parsing with U-DOP. *CoNLL '06*.

Rens Bod, 2007. Is the end of supervised parsing in sight? *ACL '07*.

Leo Breiman, 1996. Bagging predictors. *Machine Learning*, 24(2):123–140.

Rich Caruana and Alexandru Niculescu-Mizil, 2006. An empirical comparison of supervised learning algorithms. *ICML '06*.

Alexander Clark, 2001. *Unsupervised language acquisition: theory and practice*. Ph.D. thesis, University of Sussex.

Alexander Clark, 2003. Combining distributional and morphological information for part of speech induction. *EACL '03*.

Shay Cohen, Kevin Gimpel and Noah Smith, 2008. Logistic normal priors for unsupervised probabilistic grammar induction. *NIPS '08*.

Shay Cohen and Noah Smith, 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. *NAACL '09*.

David Cohn, Les Atlas and Richard Ladner. 1994. Improving generalization with active learning. *Machine Learning*, 15(2):201–221.

Simon Dennis, 2005. An exemplar-based approach to unsupervised parsing. *CogSci '05*.

W. N. Francis and H. Kucera 1979. Manual of information to accompany a standard corpus of present-day edited American English, for use with digital computers. *Department of Linguistics, Brown University Press, Providence, RI*.

Yoav Freund and Robert E. Schapire, 1996. Experiments with a new boosting algorithm. *ICML '96*.

William Headden III, Mark Johnson and David McClosky, 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. *NAACL '09*.

John Henderson and Eric Brill, 2000. Bagging and boosting a treebank parser. *NAACL '00*.

Rebecca Hwa. 2004. Sample selection for statistical parsing. *Computational Linguistics*, 30(3):253–276.

Daisuke Kawahara and Kiyotaka Uchimoto 2008. Learning reliability of parses for domain adaptation of dependency parsing. *IJCNLP '08*.

Dan Klein and Christopher Manning, 2002. A generative constituent-context model for improved grammar induction. *ACL '02*.

Dan Klein and Christopher Manning, 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. *ACL '04*.

Dan Klein, 2005. *The unsupervised learning of natural language structure.* Ph.D. thesis, Stanford University.

Jin–Dong Kim, Tomoko Ohta, Yuka Teteisi and Jun'ichi Tsujii, 2003. GENIA corpus – a semantically annotated corpus for bio-textmining. *Bioinformatics*, (supplement: 11th ISMB) 19:i180–i182, Oxford University Press, 2003.

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

David McClosky, Eugene Charniak, and Mark Johnson, 2006. Reranking and self-training for parser adaptation. *ACL-COLING '06*.

Sujith Ravi, Kevin Knight and Radu Soricut, 2008. Automatic prediction of parser accuracy. *EMNLP '08*.

Roi Reichart and Ari Rappoport, 2007. An ensemble method for selection of high quality parses. *ACL '07*.

Roi Reichart and Ari Rappoport, 2009a. Sample selection for statistical parsers: cognitively driven algorithms and evaluation measures. *CoNLL '09*.

Roi Reichart and Ari Rappoport, 2009b. Automatic selection of high quality parses created by a fully unsupervised parser. *CoNLL '09*.

Yoav Seginer, 2007. Fast unsupervised incremental parsing. *ACL '07*.

Noah Smith and Jason Eisner, 2006. Annealing structural bias in multilingual weighted grammar induction. *ACL-COLING '06*.

Valentin Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky, 2010. From baby steps to leapfrog: how "less is more" in unsupervised dependency parsing. *NAACL '10*.

Ricardo Vilalta and Irina Rish, 2003. A decomposition of classes via clustering to explain and improve naive bayes. *ECML '03*.

Alexander Yates, Stefan Schoenmackers and Oren Etzioni, 2006. Detecting parser errors using web-based semantic filters . *EMNLP '06*.

Unsupervised Parse Selection for HPSG

Rebecca Dridan and **Timothy Baldwin**
Dept. of Computer Science and Software Engineering
University of Melbourne, Australia
rdridan@csse.unimelb.edu.au, tb@ldwin.net

Abstract

Parser disambiguation with precision grammars generally takes place via statistical ranking of the parse yield of the grammar using a supervised parse selection model. In the standard process, the parse selection model is trained over a hand-disambiguated treebank, meaning that without a significant investment of effort to produce the treebank, parse selection is not possible. Furthermore, as treebanking is generally streamlined with parse selection models, creating the initial treebank without a model requires more resources than subsequent treebanks. In this work, we show that, by taking advantage of the constrained nature of these HPSG grammars, we can learn a discriminative parse selection model from raw text in a purely unsupervised fashion. This allows us to bootstrap the treebanking process and provide better parsers faster, and with less resources.

1 Introduction

Parsing with precision grammars is generally a two-stage process: (1) the full parse yield of the precision grammar is calculated for a given item, often in the form of a packed forest for efficiency (Oepen and Carroll, 2000; Zhang et al., 2007); and (2) the individual analyses in the parse forest are ranked using a statistical model ("parse selection"). In the domain of treebank parsing, the Charniak and Johnson (2005) reranking parser adopts an analogous strategy, except that ranking and pruning are incorporated into the first stage, and the second stage is based on only the top-ranked parses from the first stage. For both styles of parsing, however, parse selection is based on a statistical model learned from a pre-existing treebank associated with the grammar. Our interest in this paper is in completely removing this requirement of parse selection on explicitly treebanked data, ie the development of fully unsupervised parse selection models.

The particular style of precision grammar we experiment with in this paper is HPSG (Pollard and Sag, 1994), in the form of the DELPH-IN suite of grammars (`http://www.delph-in.net/`). One of the main focuses of the DELPH-IN collaboration effort is multilinguality. To this end, the Grammar Matrix project (Bender et al., 2002) has been developed which, through a set of questionnaires, allows grammar engineers to quickly produce a core grammar for a language of their choice. Bender (2008) showed that by using and expanding on this core grammar, she was able to produce a broad-coverage precision grammar of Wambaya in a very short amount of time. However, the Grammar Matrix can only help with the first stage of parsing. The statistical model used in the second stage of parsing (ie parse selection) requires a treebank to learn the features, but as we explain in Section 2, the treebanks are created by parsing, preferably *with* a statistical model. In this work, we look at methods for bootstrapping the production of these statistical models without having an annotated treebank. Since many of the languages that people are building new grammars for are under-resourced, we can't depend on having any external information or NLP tools, and so the methods we examine are purely unsupervised, using nothing more than the grammars them-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 694–704,
MIT, Massachusetts, USA, 9-11 October 2010.

selves and raw text. We find that, not only can we produce models that are suitable for kick-starting the treebanking process, but the accuracy of these models is comparable to parsers trained on gold standard data (Clark and Curran, 2007b; Miyao and Tsujii, 2008), which have been successfully used in applications (Miyao et al., 2008).

2 The problem

The current method of training a parse selection model uses the [incr tsdb()] treebanking mechanism (Oepen, 2001) and works well for updating models for mature grammars, although even for these grammars, building a new model for a different domain requires a time-consuming initial treebanking effort. The treebanks used with DELPH-IN grammars are *dynamic* treebanks (Oepen et al., 2004) created by parsing text and having an annotator select the correct analysis (or discard all of them). The annotation process involves making binary decisions based on so-called parse discriminants (Carter, 1997). Whenever the grammar is changed, the treebank can be quickly updated by re-parsing and re-applying the old annotation decisions. This treebanking process not only produces gold standard trees, but also a set of non-gold trees which provides the negative training data necessary for a discriminative maximum entropy model.

The standard process for creating a parse selection model is:

1. parse the training set, recording up to 500 highest-ranking parses for each sentence;

2. treebank the training set;

3. extract features from the gold and non-gold parses;

4. learn feature weights using the TADM toolkit.[1] (Malouf, 2002)

The useful training data from this process is the parses from those sentences for which: more than one parse was found; and at least one parse has been annotated as correct. That is, there needs to be both gold and non-gold trees for any sentence to be used in training the discriminative model.

There are two issues with this process for new grammars. Firstly, treebanking takes many person-hours, and is hence both time-consuming and expensive. Complicating that is the second issue: N-best parsing requires a statistical model. While it is possible to parse exhaustively with no model, parsing is much slower, since the unpacking of results is time-consuming. Selective unpacking (Zhang et al., 2007) speeds this up a great deal, but requires a parse selection model. Treebanking is also much slower when the parser must be run exhaustively, since there are usually many more analyses to manually discard.

This work hopes to alleviate both problems. By producing a statistical model without requiring human treebanking, we can have a working and efficient parser with less human effort. Even if the top-1 parses this parser produces are not as accurate as those trained on gold standard data, this model can be used to produce the N-best analyses for the treebanker. Since our models are much better than random selection, we can afford to reduce N and still have a reasonable likelihood that the correct parse is in that top N, making the job of the treebanker much faster, and potentially leading to even better parse selection accuracy based on semi-supervised or fully-supervised parse selection.

3 Data and evaluation

Our ultimate goal is to use these methods for under-resourced languages but, since there are no pre-existing treebanks for these languages, we have no means to measure which method produces the best results. Hence, in this work, we experiment with languages and grammars where we have gold standard data, in order to be able to evaluate the quality of the parse selection models. Since we have gold-standard trained models to compare with, this enables us to fully explore how these unsupervised methods work, and show which methods are worth trying in the more time-consuming and resource-intensive future experiments on other languages. It is worth reinforcing that the gold-standard data is used for evaluation only, except in calculating the supervised parse selection accuracy as an upper-bound.

The English Resource Grammar (ERG:

[1] http://tadm.sourceforge.net/

Language	Sentences	Average words	Average parses
Japanese	6769	10.5	49.6
English	4855	9.0	59.5

Table 1: Initial model training data, showing the average word length per sentence, and also the ambiguity measured as the average number of parses found per sentence.

Flickinger (2002)) is an HPSG-based grammar of English that has been under development for many person years. In order to examine the cross-lingual applicability of our methods, we also use Jacy, an HPSG-based grammar of Japanese (Siegel and Bender, 2002). In both cases, we use grammar versions from the "Barcelona" release, from mid-2009.

3.1 Training Data

Both of our grammars come with statistical models, and the parsed data and gold standard annotations used to create these models are freely available. As we are trying to simulate a fully unsupervised setup, we didn't want any influence from these earlier models. Hence, in our experiments we used the parsed data from those sentences that received less than 500 parses and ignored any ranking, thus annulling the effects of the statistical model. This led to a reduced data set, both in the number of sentences, and in the fact that the more ambiguous sentences were discarded, but it allows clean comparison between different methods, without incorporating external information. The details of our training sets are shown in Table 1,[2] indicating that the sentence lengths are relatively short, and hence the ambiguity (measured as average parses per sentence) is low for both our grammars. The ambiguity figures also suggest that the Japanese grammar is more constrained (less ambiguous) than the English grammar, since there are, on average, more parses per sentence for English, even with a lower average sentence length.

3.2 Test Data

The test data sets used throughout our experiments are described in Table 2. The *tc-006* data set is from the same Tanaka Corpus (Tanaka, 2001) which was used for the Japanese training data. There is a wider variety of treebanked data available for the English grammar than for the Japanese. We use the *jhpstg$_t$* data set, which consists of text from Norwegian tourism brochures, from the same LOGON corpus as the English training data (Oepen et al., 2004). In order to have some idea of domain effects, we also use the *catb* data set, the text of an essay on open-source development.[3] We see here that the sentences are longer, particularly for the English data. Also, since we are not artificially limiting the parse ambiguity by ignoring those with 500 or more parses, the ambiguity is much higher. This ambiguity figure gives some indication of the difficulty of the parse selection task. Again we see that the English sentences are more ambiguous, much more in this case, making the parse selection task difficult. In fact, the English ambiguity figures are an under-estimate, since some of the longer sentences timed out before producing a parse count. This ambiguity can be a function of the sentence length or the language itself, but also of the grammar. A more detailed and informative grammar makes more distinctions, not all of which are relevant for every analysis.

Test Set	Language	Sentences	Average words	Average parses
tc-006	Japanese	904	10.7	383.9
jhpstg$_t$	English	748	12.8	4115.1
catb	English	534	17.6	9427.3

Table 2: Test data, showing the average word length per sentence, and also the ambiguity measured as the average number of parses found per sentence. Note that the ambiguity figures for the English test sets are under-estimates, since some of the longer sentences timed out before giving an analysis count.

3.3 Evaluation

The exact match metric is the most common accuracy metric used in work with the DELPH-IN tool set, and refers to the percentage of sentences for which the top parse matched the gold parse in every way. This is akin to the sentence accuracy that is occasionally reported in the parsing literature, except

[2] Any sentences that do not have both gold and non-gold analyses (ie, had no correct parse, only one parse, or none) are not included in these figures.

[3] The Cathedral and the Bazaar, by Eric Raymond. Available from: `http://catb.org/esr/writings/cathedral-bazaar/`

that it also includes fine-grained syntactico-semantic features that are not often present in other parsing frameworks. Exact match is a useful metric for parse selection evaluation, but it is very blunt-edged, and gives no way of evaluating *how close* the top parse was to the gold standard. Since these are very detailed analyses, it is possible to get one detail wrong and still have a useful analysis. Hence, in addition to exact match, we also use the EDM$_{NA}$ evaluation defined by Dridan (2009). This is a predicate–argument style evaluation, based on the semantic output of the parser (MRS: Minimal Recursion Semantics (Copestake et al., 2005)). This metric is broadly comparable to the predicate–argument dependencies of CCGBank (Hockenmaier and Steedman, 2007) or of the ENJU grammar (Miyao and Tsujii, 2008), and also somewhat similar to the grammatical relations (GR) of the Briscoe and Carroll (2006) version of DepBank. The EDM$_{NA}$ metric matches triples consisting of predicate names and the argument type that connects them.[4]

4 Initial Experiments

All of our experiments are based on the same basic process: (1) for each sentence in the training data described in Section 3.1, label a subset of analyses as correct and the remainder as incorrect; (2) train a model using the same features and learner as in the standard process of Section 2; (3) parse the test data using that model; and (4) evaluate the accuracy of the top analyses. The differences lay in how the 'correct' analyses are selected each time. Each of the following sections detail different methods for nominating which of the (up to 500) analyses from the training data should be considered pseudo-gold for training the parse selection model.

4.1 Upperbound and baseline models

As a first step we evaluated each data set using an upperbound and a baseline model. The upperbound model in this case is the model trained with gold standard annotations. These accuracy figures are slightly lower than others found in the literature for this data, since, to allow for comparison, we limited the training data to the sets described in Table 1. By throwing out those sentences with more than 500 parses, we exclude much of the data that is used in the standard model and so our exact match figures are slightly lower than might be expected.

Test Set	**Exact**	**EDM**		
	Match	**Precision**	**Recall**	**F-score**
tc-006	72.90	0.961	0.957	0.959
jhpstg$_t$	48.07	0.912	0.908	0.910
catb	22.29	0.838	0.839	0.839

Table 3: Accuracy of the gold standard-based parse selection model.

Test Set	**Exact**	**EDM**		
	Match	**Precision**	**Recall**	**F-score**
tc-006	17.26	0.779	0.839	0.807
jhpstg$_t$	12.48	0.720	0.748	0.734
catb	8.30	0.646	0.698	0.671

Table 4: Accuracy of the baseline model, trained on randomly selected pseudo-gold analyses.

For the baseline model, we used random selection to select our gold analyses. For this experiment, we randomly assigned one parse from each sentence in the training data to be correct (and the remainder of analyses as incorrect), and then used that 'gold standard' to train the model. Results for the upperbound and baseline models are shown in Tables 3 and 4.

As expected, the results for Japanese are much higher, since the lower ambiguity makes this an easier task. The *catb* test set results suffer, not only from being longer, more ambiguous sentences, but also because it is completely out of the domain of the training data.

The exact match results from the random baseline are approximately what one might expect, given the respective ambiguity levels in Table 2. The EDM figures are perhaps higher than might be expected given random selection from the entire parse forest. This results from using a precision grammar, with an inbuilt notion of grammaticality, hence constraining the parser to only produce somewhat reasonable parses, and creating a reasonably high baseline for our parse selection experiments.

We also tried a separate baseline, eliminating the parse selection model altogether, and using random selection directly to select the top analysis. The exact match and EDM precision results were slightly lower than using random selection to train a model,

[4] The full EDM metric also includes features such as tense and aspect, but this is less comparable to the other metrics mentioned.

which may be due to the learner giving weight to features that are common across the training data, but the differences weren't significant. Recall was significantly lower when using random selection directly, due to the time outs caused by running without a model. For this reason, we use the random selection-based model results as our baseline for the other unsupervised parse selection models, noting that correctly identifying approximately three quarters of the dependencies in the *jhpstg*$_t$ set, and over 80% when using the Japanese grammar, is a fairly high baseline.

4.2 First attempts

As a first approach to unsupervised parse selection, we looked at two heuristics to designate some number of the analyses as 'gold' for training. Both of these heuristics looked independently at the parses of each sentence, rather than calculating any numbers across the whole training set.

The first method builds on the observation from the random selection-based model baseline experiment that just giving weight to common features could improve parser accuracy. In this case, we looked at the edges of the parsing chart. For each sentence, we counted the number of times an edge was present in an analysis, and used that number (normalised by the total number of times any edge was used) as the *edge weight*. We then calculated an analysis score by summing the edge weights of all the edges in that analysis, and dividing by the number of edges, to give an average edge weight for an analysis. All analyses that had the best analysis score for a sentence were designated 'gold'. Since it was possible for multiple analyses to have the same score, there could be multiple gold analyses for any one sentence. If all the analyses had the same score, this sentence could not be used as part of the training data. This method has the effect of selecting the parse(s) most like all the others, by some definitions the *centroid* of the parse forest. This has some relationship to the partial training method described by Clark and Curran (2006), where the most frequent dependencies where used to train a model for the C&C CCG parser. In that case, however, the dependencies were extracted only from analyses that matched the gold standard supertag sequence, rather than the whole parse forest.

Test Set	**Exact Match**		**F-score**	
	Edges	**Branching**	**Edges**	**Branching**
tc-006	17.48	21.35	0.815	0.822
jhpstg$_t$	15.27	17.53	0.766	0.780
catb	9.36	10.86	0.713	0.712

Table 5: Accuracy for each test set, measured both as percentage of sentences that exactly matched the gold standard, and f-score over elementary dependencies.

The second heuristic we tried is one often used as a baseline method: degree of right (or left) branching. In this instance, we calculated the degree of branching as the number of right branches in a parse divided by the number of left branches (and vice versa for Japanese, a predominantly left-branching language). In the same way as above, we designated all parses with the best branching score as 'gold'. Again, this is not fully discriminatory, and it was common to get multiple gold trees for a given sentence.

Table 5 shows the results for these two methods. All the results show an improvement over the baseline, with all but the F-score for the Edges method of *tc-006* being at a level of statistical significance.[5] The only statistically significant difference between the Edges and Branching methods is over the *jhpstg*$_t$ data set. While improvement over random is encouraging, the results were still uninspiring and so we moved on to slightly more complex methods, described in the next section.

5 Supertagging Experiments

The term *supertags* was first used by Bangalore and Joshi (1999) to describe fine-grained part of speech tags which include some structural or dependency information. In that original work, the supertags were LTAG (Schabes and Joshi, 1991) elementary trees, and they were used for the purpose of speeding up parsing by restricting the allowable leaf types. Subsequent work involving supertags has mostly focussed on this efficiency goal, but they can also be used to inform parse selection. Dalrymple (2006) and Blunsom (2007) both look at how discriminatory a tag sequence is in filtering a parse forest. This

[5] All statistical significance tests in these experiments use the computationally-intensive randomisation test described in Yeh (2000), with $p < 0.05$.

work has shown that tag sequences can be successfully used to restrict the set of parses produced, but generally are not discriminatory enough to distinguish a single best parse. Toutanova et al. (2002) present a similar exploration but also go on to include probabilities from a HMM model into the parse selection model as features. There has also been some work on using lexical probabilities for domain adaptation of a model (Hara et al., 2007; Rimell and Clark, 2008). In Dridan (2009), tag sequences from a supertagger are used together with other factors to re-rank the top 500 parses from the same parser and English grammar we use in this research, and achieve some improvement in the ranking where tagger accuracy is sufficiently high. We use a similar method, one level removed, in that we use the tag sequences to select the 'gold' parse(s) that are then used to train a model, as in the previous sections.

5.1 Gold Supertags

In order to test the viability of this method, we first experimented using gold standard tags, extracted from the gold standard parses. Supertags come in many forms, depending on both the grammar formalism and the implementation. For this work, we use HPSG lexical types (lextypes), the native word classes in the grammars. These lextypes encode part of speech and subcategorisation information, as well as some more idiosyncratic features of words, such as restrictions on preposition forms, mass/count distinctions and comparative versus superlative forms of adjectives. As a few examples from the English grammar, `v_np_le` represents a basic transitive verb, while `n_pp_c-of_le` represents a count noun that optionally takes a prepositional phrase complement headed by *of*. The full definition of a lextype consists of a many-featured AVM (attribute value matrix), but the type names have been deliberately chosen to represent the main features of each type. In the Dridan (2009) work, parse ranking showed some improvement when morphological information was added to the tags. Hence, we also look at more fine-grained tags constructed by concatenating appropriate morphological rules onto the lextypes, as in `v_np_le:past_verb_orule` (ie a simple transitive verb with past tense).

We used these tags by extracting the tag sequence from the leaf types of all the parses in the forest, marking as 'gold' any parse that had the same sequence as the gold standard parse and then training the models as before. Table 6 shows the results from parsing with models based on both the basic lextype and the lextype with morphology. The results are promising. They still fall well below training purely on gold standard data (at least for the in-domain sets), since the tag sequences are not fully discriminatory and hence noise can creep in, but accuracy is significantly better than the heuristic methods tried earlier. This suggested that, at least with a reasonably accurate tagger, this was a viable strategy for training a model. With no significant difference between the basic and +morph versions of the tag set, we decided to use the basic lextypes as tags, since a smaller tag set should be easier to tag with. However, we first had to train a tagger, without using any gold standard data.

Test Set	Exact Match		F-score	
	lextype	+morph	lextype	+morph
tc-006	40.49	41.37	0.903	0.903
jhpstg$_t$	32.93	32.93	0.862	0.858
catb	20.41	19.85	0.798	0.794

Table 6: Accuracy using gold tag sequence compatibility to select the 'gold' parse(s).

5.2 Unsupervised Supertagging

Research into unsupervised part-of-speech tagging with a tag dictionary (sometimes called weakly supervised POS tagging) has been going on for many years (cf Merialdo (1994), Brill (1995)), but generally using a fairly small tag set. The only work we know of on unsupervised tagging for the more complex *supertags* is from Baldridge (2008), and more recently, Ravi et al. (2010a). In this work, the constraining nature of the (CCG) grammar is used to mitigate the problem of having a much more ambiguous tag set. Our method has a similar underlying idea, but the implementation differs both in the way we extract the word-to-tag mappings, and also how we extract and use the information from the grammar to initialise the tagger model.

We chose to use a simple first-order Hidden Markov Model (HMM) tagger, using the implementation of Dekang Lin,[6] which re-estimates probabilities, given an initial model, using the Baum-Welch variant of the Expectation-Maximisation (EM) algorithm. One possibility for an initial model was to extract the word-to-lextype mappings from the grammar lexicon as Baldridge does, and make all starting probabilities uniform. However, our lexicon maps between lextypes and *lemmas*, rather than inflected word forms, which is what we'd be tagging. That is to say, from the lexicon we could learn that the lemma *walk* can be tagged as `v_pp*_dir_le`, but we could not directly extract the fact that therefore *walked* should also receive that tag.[7] For this reason, we decided it would be simplest to initialise our probability estimates using the output of the parser, feeding in only those tag sequences which are compatible with analyses in the parse forest for that item. This method takes advantage of the fact that, because the grammars are heavily constrained, the parse forest only contains viable tag sequences. Since parsing without a model is slow, we restricted the training set to those sentences shorter than a specific word length (12 for English and 15 for Japanese, since that was the less ambiguous grammar and hence faster).

Table 7 shows how much parsed data this gave us. From this parsed data we extracted tag-to-word and tag-to-tag frequency counts from all parses for all sentences, and used these frequencies to produce the emission and transition probabilities, respectively. The emission probabilities were taken directly from the normalised frequency counts, but for the transition probabilities we allow for all possible transitions, and add one to all counts before normalising. This model we call our *initial counts* model. The *EM trained* model is then produced by starting with this initial model and running the Baum-Welch algorithm using raw text sentences from the training corpus.

5.3 Supertagging-based parse selection models

We use both the *initial counts* and *EM trained* models to tag the training data from Table 1 and then compared this with the extracted tag sequences used in the gold tag experiment. Since we could no longer assume that our tag sequence would be present within the extracted tag sequences, we used the percentage of tokens from a parse whose lextype matched our tagged sequence as the parse score. Again, we marked as 'gold' any parse that had the best parse score for each sentence, and trained a new parse selection model.

	Japanese	**English**
Parsed Sentences	9760	3770
Average Length	9.63	6.36
Average Parses	80.77	96.29
Raw Sentences	13500	9410
Raw Total Words	146053	151906

Table 7: Training data for the HMM tagger (both the parsed data from which the initial probabilities were derived, and the raw data which was used to estimated the *EM trained* models).

	Exact Match		**F-score**	
Test Set	**Initial counts**	**EM trained**	**Initial counts**	**EM trained**
tc-006	32.85	40.38	0.888	0.898
jhpstg$_t$	26.29	24.04	0.831	0.827
catb	14.61	14.42	0.782	0.783

Table 8: Accuracy using tag sequences from a HMM tagger to select the 'correct' parse(s). The *initial counts* model was based on using counts from a parse forest to approximate the emission and transition probabilities. The *EM trained* model used the Baum Welch algorithm to estimate the probabilities, starting from the initial counts state.

Table 8 shows the results of parsing with these models. The results are impressive, significantly higher than all our previous unsupervised methods.

Interestingly, we note that there is no significant difference between the *initial count* and *EM trained* models for the English data. To explore why this might be so, we looked at the tagger accuracy for both models over the respective training data sets, shown in Table 9. The results are not conclusive. For both languages, the *EM trained* model is *less* accurate, though not significantly so for Japanese. However, this insignificant tagger accuracy decrease for Japanese produced a significant *increase* in parser accuracy, while a more pronounced tagger accuracy decrease had no significant effect on parser accuracy in English.

[6] Available from `http://webdocs.cs.ualberta.ca/~lindek/hmm.htm`

[7] Morphological processing occurs before lexicon lookup in the PET parser.

Language	Initial counts	EM trained
Japanese	84.4	83.3
English	71.7	64.6

Table 9: Tagger accuracy over the training data, using both the *initial counts* and the *EM trained* models.

There is much potential for further work in this direction, experimenting with more training data or more estimation iterations, or even looking at different estimators as suggested in Johnson (2007) and Ravi et al. (2010b). There is also the issue of whether tag accuracy is the best measure for indicating potential parse accuracy. The Japanese parsing results are already equivalent to those achieved using gold standard tags. It is possible that parsing accuracy is reasonably insensitive to tagger accuracy, but it is also possible that there is a better metric to look at, such as tag accuracy over frequently confused tags.

6 Discussion

The results of Table 8 show that, using no human annotated data, we can get exact match results that are almost half way between our random baseline and our gold-standard-trained upperbound. EDM F-scores of 90% and 83% over in-domain data compare well with dependency-based scores from other parsers, although a direct comparison is very difficult to do (Clark and Curran, 2007a; Miyao et al., 2007). It still remains to see whether this level of accuracy is good enough to be useful. The main aim of this work is to bootstrap the treebanking process for new grammars, but to conclusively show the efficacy of our methods in that situation requires a long-term experiment that we are now starting, based on the results we have here. Another possible use for these methods was alluded to in Section 2: producing a new model for a new domain.

Results at every stage have been much worse for the *catb* data set, compared to the other jhpstg_t English data set. While sentence length plays some part, the major reason for this discrepancy was domain mismatch between the training and test data. One method that has been successfully used for domain adaption in parsing is self-training (McClosky et al., 2006). In this process, data from the new domain is parsed with the parser trained on the old domain, and then the top analyses of the parsed new domain data are added to the training data, and the parser is re-trained. This is generally considered a semi-supervised method, since the original parser is trained on gold standard data. In our case, we wanted to test whether parsing data from the new domain using our *unsupervised* parse selection model was accurate enough to still get an improvement using self-training for domain adaptation.

Source of ‘Gold’ Data	Exact Match	F-score
Random Selection	8.30	0.671
Supertags (*initial counts*)	14.61	0.782
Gold Standard	22.29	0.839
Self-training	15.92	0.791

Table 10: Accuracy results over the out-of-domain *catb* data set, using the *initial counts* unsupervised model to produce in-domain training data in a self-training set up. The previous results are shown for easy comparison.

It is not immediately clear what one might consider to be the ‘domain’ of the *catb* test set, since domain is generally very vaguely defined. In this case, there was a limited amount of text available from other essays by the same author.[8] While the topics of these essays vary, they all relate to the social side of technical communities, and so we used this to represent in-domain data for the *catb* test set. It is, however, a fairly small amount of data for self-training, being only around 1000 sentences. We added the results of parsing this data to the training set we used to create the *initial counts* model and again retrained and parsed. Table 10 shows the results. Previous results for the *catb* data set are given for comparison.

The results show that the completely unsupervised parse selection method produces a top parse that is at least accurate enough to be used in self-training, providing a cheap means of domain adaptation. In future work, we hope to explore this avenue of research further.

7 Conclusions and Further Work

Comparing Tables 8 and 4, we can see that for both English and Japanese, we are able to achieve parse selection accuracy well above our baseline of a ran-

[8] `http://www.catb.org/esr/writings/homesteading/`

dom selection-based model using only the information available in the grammar and raw text. This was in part because it is possible to extract a reasonable tagging model from uncorrected parse data, due to the constrained nature of these grammars. These models will hopefully allow grammar engineers to more easily build statistical models for new languages, using nothing more than their new grammar and raw text.

Since fully evaluating the potential for building models for new *languages* is a long-term ongoing experiment, we looked at a more short-term evaluation of our unsupervised parse selection methods: building models for new *domains*. A preliminary self-training experiment, using our *initial counts* tagger trained model as the starting point, showed promising results for domain adaptation.

There are plenty of directions for further work arising from these results. The issues surrounding what makes a good tagger for this purpose, and how can we best learn one without gold training data, would be one possibly fruitful avenue for further exploration. Another interesting slant would be to investigate domain effects of the tagger. Previous work has already found that training just a lexical model on a new domain can improve parsing results. Since the optimal tagger 'training' we saw here (for English) was merely to read off frequency counts for parsed data, it would be easy to retrain the tagger on different domains. Alternatively, it would be interesting so see how much difference it makes to train the tagger on one set of data, and use that to tag a model training set from a different domain. Other methods of incorporating the tagger output could also be investigated. Finally, a user study involving a grammar engineer working on a new language would be useful to validate the results we found here and confirm whether they are indeed helpful in bootstrapping a new grammar.

Acknowledgements

This research was supported by Australian Research Council grant no. DP0988242 and Microsoft Research Asia.

References

Jason Baldridge. 2008. Weakly supervised supertagging with grammar-informed initialization. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 57–64, Manchester, UK.

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: an approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Emily M. Bender, Dan Flickinger, and Stephan Oepen. 2002. The grammar matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. In *Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics*, pages 8–14, Taipei, Taiwan.

Emily M. Bender. 2008. Evaluating a crosslinguistic grammar resource: A case study of Wambaya. In *Proceedings of the 46th Annual Meeting of the ACL*, pages 977–985, Columbus, USA.

Philip Blunsom. 2007. *Structured Classification for Multilingual Natural Language Processing*. Ph.D. thesis, Department of Computer Science and Software Engineering, the University of Melbourne.

Eric Brill. 1995. Unsupervised learning of disambiguation rules for part of speech tagging. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 1–13, Cambridge, USA.

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 and the 21st International Conference on Computational Linguistics*, pages 41–48, Sydney, Australia.

David Carter. 1997. The treebanker: a tool for supervised training of parsed corpora. In *Proceedings of a Workshop on Computational Environments for Grammar Development and Linguistic Engineering*, pages 9–15, Madrid, Spain.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 173–180, Ann Arbor, USA.

Stephen Clark and James R. Curran. 2006. Partial training for a lexicalized-grammar parser. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL (NAACL)*, pages 144–151, New York City, USA.

Stephen Clark and James R. Curran. 2007a. Formalism-independent parser evaluation with CCG and DepBank. In *Proceedings of the 45th Annual Meeting of the ACL*, 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.

Ann Copestake, Dan Flickinger, Ivan A. Sag, and Carl Pollard. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, vol 3(no 4):pp 281–332.

Mary Dalrymple. 2006. How much can part-of-speech tagging help parsing? *Natural Language Engineering*, 12(4):373–389.

Rebecca Dridan. 2009. *Using lexical statistics to improve HPSG parsing*. Ph.D. thesis, Saarland University.

Dan Flickinger. 2002. On building a more efficient grammar by exploiting types. In Stephan Oepen, Dan Flickinger, Jun'ichi Tsujii, and Hans Uszkoreit, editors, *Collaborative Language Engineering*, pages 1–17. Stanford: CSLI Publications.

Tadayoshi Hara, Yusuke Miyao, and Jun'ichi Tsujii. 2007. Evaluating impact of re-training a lexical disambiguation model on domain adaptation of an HPSG parser. In *Proceedings of the 10th International Conference on Parsing Technology (IWPT 2007)*, pages 11–22, Prague, Czech Republic.

Julia Hockenmaier and Mark Steedman. 2007. CCG-bank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396, September.

Mark Johnson. 2007. Why doesnt 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, Prague, Czech Republic.

Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th Conference on Natural Language Learning*, Taipei, Taiwan.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL*, pages 152–159, New York City, USA.

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

Yusuke Miyao and Jun'ichi Tsujii. 2008. Feature forest models for probabilistic HPSG parsing. *Computational Linguistics*, 34(1):35–80.

Yusuke Miyao, Kenji Sagae, and Jun'ichi Tsujii. 2007. Towards framework-independent evaluation of deep linguistic parsers. In *Proceedings of the GEAF 2007 Workshop*, Palo Alto, California.

Yusuke Miyao, Rune Sætre, Kenji Sagae, Takuya Matsuzaki, and Jun'ichi Tsujii. 2008. Task-oriented evaluation of syntactic parsers and their representations. In *Proceedings of the 46th Annual Meeting of the ACL*, pages 46–54, Columbus, USA.

Stephan Oepen and John Carroll. 2000. Ambiguity packing in constraint-based parsing - practical results. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics*, pages 162–169, Seattle, USA.

Stephan Oepen, Dan Flickinger, Kristina Toutanova, and Christopher D. Manning. 2004. LinGO redwoods. a rich and dynamic treebank for HPSG. *Journal of Research in Language and Computation*, 2(4):575–596.

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.

Sujith Ravi, Jason Baldridge, and Kevin Knight. 2010a. Minimized models and grammar-informed initialization for supertagging with highly ambiguous lexicons. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 495–503, Uppsala, Sweden.

Sujith Ravi, Ashish Vaswani, Kevin Knight, and David Chiang. 2010b. Fast, greedy model minimization for unsupervised tagging. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 940–948, Beijing, China.

Laura Rimell and Stephen Clark. 2008. Adapting a lexicalized-grammar parser to contrasting domains. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*, pages 475–484, Honolulu, USA.

Yves Schabes and Aravind K. Joshi. 1991. Parsing with lexicalized tree adjoining grammar. In Masaru Tomita, editor, *Current Issues in Parsing Technology*, chapter 3, pages 25–48. Kluwer.

Melanie Siegel and Emily M. Bender. 2002. Efficient deep processing of japanese. In *Proceedings of the 3rd Workshop on Asian Language Resources and International Standardization. Coling 2002 Post-Conference Workshop.*, Taipei, Taiwan.

Yasuhito Tanaka. 2001. Compilation of a multilingual parallel corpus. In *Proceedings of PACLING 2001*, pages 265–268, Kitakyushu, Japan.

Kristina Toutanova, Chistopher D. Manning, Stuart M. Shieber, Dan Flickinger, and Stephan Oepen. 2002. Parse disambiguation for a rich HPSG grammar. In *First Workshop on Treebanks and Linguistic Theories (TLT2002)*, pages 253–263.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, pages 947–953, Saarbrcken, Germany.

Yi Zhang, Stephan Oepen, and John Carroll. 2007. Efficiency in unification-based n-best parsing. In *Proceedings of the 10th international conference on parsing technologies (IWPT 2007)*, pages 48–59, Prague, Czech Republic.

Uptraining for Accurate Deterministic Question Parsing

Slav Petrov, Pi-Chuan Chang, Michael Ringgaard, Hiyan Alshawi
Google Research
{slav,pichuan,ringgaard,hiyan}@google.com

Abstract

It is well known that parsing accuracies drop significantly on out-of-domain data. What is less known is that some parsers suffer more from domain shifts than others. We show that dependency parsers have more difficulty parsing questions than constituency parsers. In particular, deterministic shift-reduce dependency parsers, which are of highest interest for practical applications because of their linear running time, drop to 60% labeled accuracy on a question test set. We propose an *uptraining* procedure in which a deterministic parser is trained on the output of a more accurate, but slower, latent variable constituency parser (converted to dependencies). Uptraining with 100K unlabeled questions achieves results comparable to having 2K labeled questions for training. With 100K unlabeled and 2K labeled questions, uptraining is able to improve parsing accuracy to 84%, closing the gap between in-domain and out-of-domain performance.

1 Introduction

Parsing accuracies on the popular Section 23 of the Wall Street Journal (WSJ) portion of the Penn Treebank have been steadily improving over the past decade. At this point, we have many different parsing models that reach and even surpass 90% dependency or constituency accuracy on this test set (McDonald et al., 2006; Nivre et al., 2007; Charniak and Johnson, 2005; Petrov et al., 2006; Carreras et al., 2008; Koo and Collins, 2010). Quite impressively, models based on deterministic shift-reduce parsing algorithms are able to rival the other computationally more expensive models (see Nivre (2008) and references therein for more details). Their linear running time makes them ideal candidates for large scale text processing, and our model of choice for this paper.

Unfortunately, the parsing accuracies of all models have been reported to drop significantly on out-of-domain test sets, due to shifts in vocabulary and grammar usage (Gildea, 2001; McClosky et al., 2006b; Foster, 2010). In this paper, we focus our attention on the task of parsing questions. Questions pose interesting challenges for WSJ-trained parsers because they are heavily underrepresented in the training data (there are only 334 questions among the 39,832 training sentences). At the same time, questions are of particular interest for user facing applications like question answering or web search, which necessitate parsers that can process questions in a fast and accurate manner.

We start our investigation in Section 3 by training several state-of-the-art (dependency and constituency) parsers on the standard WSJ training set. When evaluated on a question corpus, we observe dramatic accuracy drops exceeding 20% for the deterministic shift-reduce parsers. In general, dependency parsers (McDonald et al., 2006; Nivre et al., 2007), seem to suffer more from this domain change than constituency parsers (Charniak and Johnson, 2005; Petrov et al., 2006). Overall, the latent variable approach of Petrov et al. (2006) appears to generalize best to this new domain, losing only about 5%. Unfortunately, the parsers that generalize better to this new domain have time complexities that are cubic in the sentence length (or even higher), rendering them impractical for web-scale text processing.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 705–713,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

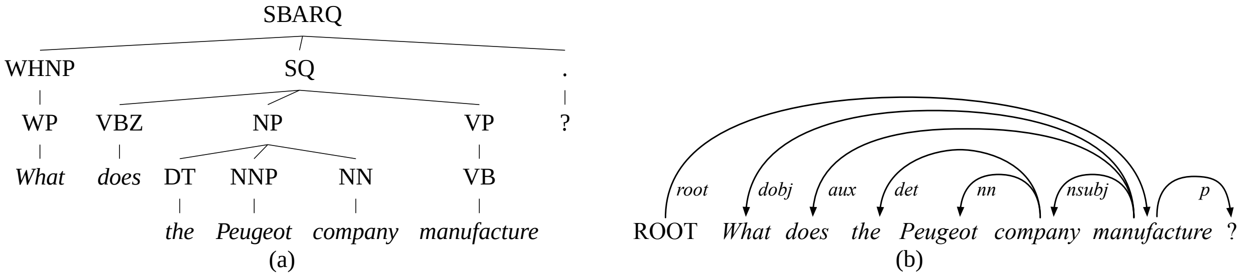


Figure 1: Example constituency tree from the QuestionBank (a) converted to labeled Stanford dependencies (b).

We therefore propose an *uptraining* method, in which a deterministic shift-reduce parser is trained on the output of a more accurate, but slower parser (Section 4). This type of domain adaptation is reminiscent of self-training (McClosky et al., 2006a; Huang and Harper, 2009) and co-training (Blum and Mitchell, 1998; Sagae and Lavie, 2006), except that the goal here is not to further improve the performance of the very best model. Instead, our aim is to train a computationally cheaper model (a linear time dependency parser) to match the performance of the best model (a cubic time constituency parser), resulting in a computationally efficient, yet highly accurate model.

In practice, we parse a large amount of unlabeled data from the target domain with the constituency parser of Petrov et al. (2006) and then train a deterministic dependency parser on this noisy, automatically parsed data. The accuracy of the linear time parser on a question test set goes up from 60.06% (LAS) to 76.94% after uptraining, which is comparable to adding 2,000 labeled questions to the training data. Combining uptraining with 2,000 labeled questions further improves the accuracy to 84.14%, fully recovering the drop between in-domain and out-of-domain accuracy.

We also present a detailed error analysis in Section 5, showing that the errors of the WSJ-trained model are primarily caused by sharp changes in syntactic configurations and only secondarily due to lexical shifts. Uptraining leads to large improvements across all error metrics and especially on important dependencies like subjects (nsubj).

2 Experimental Setup

We used the following experimental protocol throughout the paper.

2.1 Data

Our main training set consists of Sections 02-21 of the Wall Street Journal portion of the Penn Treebank (Marcus et al., 1993), with Section 22 serving as development set for source domain comparisons. For our target domain experiments, we evaluate on the QuestionBank (Judge et al., 2006), which includes a set of manually annotated questions from a TREC question answering task. The questions in the QuestionBank are very different from our training data in terms of grammatical constructions and vocabulary usage, making this a rather extreme case of domain-adaptation. We split the 4,000 questions contained in this corpus in three parts: the first 2,000 questions are reserved as a small target-domain training set; the remaining 2,000 questions are split in two equal parts, the first serving as development set and the second as our final test set. We report accuracies on the developments sets throughout this paper, and test only at the very end on the final test set.

We convert the trees in both treebanks from constituencies to labeled dependencies (see Figure 1) using the Stanford converter, which produces 46 types of labeled dependencies[1] (de Marneffe et al., 2006). We evaluate on both unlabeled (UAS) and labeled dependency accuracy (LAS).[2]

Additionally, we use a set of 2 million questions collected from Internet search queries as unlabeled target domain data. All user information was anonymized and only the search query string was retained. The question sample is selected at random after passing two filters that select queries that are

[1] We use the Stanford Lexicalized Parser v1.6.2.

[2] Because the QuestionBank does not contain function tags, we decided to strip off the function tags from the WSJ before conversion. The Stanford conversion only uses the -ADV and -TMP tags, and removing all function tags from the WSJ changed less than 0.2% of the labels (primarily tmod labels).

Training on WSJ Sections 02-21	Evaluating on WSJ Section 22				Evaluating on QuestionBank			
	F_1	UAS	LAS	POS	F_1	UAS	LAS	POS
Nivre et al. (2007)	—	88.42	84.89	95.00	—	74.14	62.81	88.48
McDonald et al. (2006)	—	89.47	86.43	95.00	—	80.01	67.00	88.48
Charniak (2000)	90.27	92.33	89.86	96.71	83.01	85.61	73.59	90.49
Charniak and Johnson (2005)	91.92	93.56	91.24	96.69	84.47	87.13	75.94	90.59
Petrov et al. (2006)	90.70	92.91	90.48	96.27	85.52	88.17	79.10	90.57
Petrov (2010)	92.10	93.85	91.60	96.44	86.62	88.77	79.92	91.08
Our shift-reduce parser	—	88.24	84.69	95.00	—	72.23	60.06	88.48
Our shift-reduce parser (gold POS)	—	90.51	88.53	100.00	—	78.30	68.92	100.00

Table 1: Parsing accuracies for parsers trained on newswire data and evaluated on newswire and question test sets.

similar in style to the questions in the QuestionBank: (i) the queries must start with an English function word that can be used to start a question (what, who when, how, why, can, does, etc.), and (ii) the queries have a maximum length of 160 characters.

2.2 Parsers

We use multiple publicly available parsers, as well as our own implementation of a deterministic shift-reduce parser in our experiments. The dependency parsers that we compare are the deterministic shift-reduce MaltParser (Nivre et al., 2007) and the second-order minimum spanning tree algorithm based MstParser (McDonald et al., 2006). Our shift-reduce parser is a re-implementation of the MaltParser, using a standard set of features and a linear kernel SVM for classification. We also train and evaluate the generative lexicalized parser of Charniak (2000) on its own, as well as in combination with the discriminative reranker of Charniak and Johnson (2005). Finally, we run the latent variable parser (a.k.a. BerkeleyParser) of Petrov et al. (2006), as well as the recent product of latent variable grammars version (Petrov, 2010). To facilitate comparisons between constituency and dependency parsers, we convert the output of the constituency parsers to labeled dependencies using the same procedure that is applied to the treebanks. We also report their F_1 scores for completeness.

While the constituency parsers used in our experiments view part-of-speech (POS) tagging as an integral part of parsing, the dependency parsers require the input to be tagged with a separate POS tagger. We use the TnT tagger (Brants, 2000) in our experiments, because of its efficiency and ease of use. Tagger and parser are always trained on the same data.

3 Parsing Questions

We consider two domain adaptation scenarios in this paper. In the first scenario (sometimes abbreviated as WSJ), we assume that we do not have any labeled training data from the target domain. In practice, this will always be the case when the target domain is unknown or very diverse. The second scenario (abbreviated as WSJ+QB) assumes a small amount of labeled training data from the target domain. While this might be expensive to obtain, it is certainly feasible for narrow domains (e.g. questions), or when a high parsing accuracy is really important.

3.1 No Labeled Target Domain Data

We first trained all parsers on the WSJ training set and evaluated their performance on the two domain specific evaluation sets (newswire and questions). As can be seen in the left columns of Table 1, all parsers perform very well on the WSJ development set. While there are differences in the accuracies, all scores fall within a close range. The table also confirms the commonly known fact (Yamada and Matsumoto, 2003; McDonald et al., 2005) that constituency parsers are more accurate at producing dependencies than dependency parsers (at least when the dependencies were produced by a deterministic transformation of a constituency treebank, as is the case here).

This picture changes drastically when the performance is measured on the QuestionBank development set (right columns in Table 1). As one

Evaluating on QuestionBank	Training on WSJ + QB				Training on QuestionBank			
	F_1	UAS	LAS	POS	F_1	UAS	LAS	POS
Nivre et al. (2007)	—	83.54	78.85	91.32	—	79.72	73.44	88.80
McDonald et al. (2006)	—	84.95	80.17	91.32	—	82.52	77.20	88.80
Charniak (2000)	89.40	90.30	85.01	94.17	79.70	76.69	69.69	87.84
Petrov et al. (2006)	90.96	90.98	86.90	94.01	86.62	84.09	78.92	87.56
Petrov (2010)	92.81	92.23	88.84	94.48	87.72	85.07	80.08	87.79
Our shift-reduce parser	—	83.70	78.27	91.32	—	80.44	74.29	88.80
Our shift-reduce parser (gold POS)	—	89.39	86.60	100.00	—	87.31	84.15	100.00

Table 2: Parsing accuracies for parsers trained on newswire and question data and evaluated on a question test set.

might have expected, the accuracies are significantly lower, however, the drop for some of the parsers is shocking. Most notably, the deterministic shift-reduce parsers lose almost 25% (absolute) on labeled accuracies, while the latent variable parsers lose around 12%.[3] Note also that even with gold POS tags, LAS is below 70% for our deterministic shift-reduce parser, suggesting that the drop in accuracy is primarily due to a syntactic shift rather than a lexical shift. These low accuracies are especially disturbing when one considers that the average question in the evaluation set is only nine words long and therefore potentially much less ambiguous than WSJ sentences. We will examine the main error types more carefully in Section 5.

Overall, the dependency parsers seem to suffer more from the domain change than the constituency parsers. One possible explanation is that they lack the global constraints that are enforced by the (context-free) grammars. Even though the MstParser finds the globally best spanning tree, all constraints are local. This means for example, that it is not possible to require the final parse to contain a verb (something that can be easily expressed by a top-level production of the form S → NP VP in a context free grammar). This is not a limitation of dependency parsers in general. For example, it would be easy to enforce such constraints in the Eisner (1996) algorithm or using Integer Linear Programming approaches (Riedel and Clarke, 2006; Martins et al., 2009). However, such richer modeling capacity comes with a much higher computational cost.

Looking at the constituency parsers, we observe that the lexicalized (reranking) parser of Charniak and Johnson (2005) loses more than the latent variable approach of Petrov et al. (2006). This difference doesn't seem to be a difference of generative vs. discriminative estimation. We suspect that the latent variable approach is better able to utilize the little evidence in the training data. Intuitively speaking, some of the latent variables seem to get allocated for modeling the few questions present in the training data, while the lexicalization contexts are not able to distinguish between declarative sentences and questions.

To verify this hypothesis, we conducted two additional experiments. In the first experiment, we collapsed the question specific phrasal categories SQ and SBARQ to their declarative sentence equivalents S and SBAR. When the training and test data are processed this way, the lexicalized parser loses 1.5% F_1, while the latent variable parser loses only 0.7%. It is difficult to examine the grammars, but one can speculate that some of the latent variables were used to model the question specific constructions and the model was able to re-learn the distinctions that we purposefully collapsed. In the second experiment, we removed all questions from the WSJ training set and retrained both parsers. This did not make a significant difference when evaluating on the WSJ development set, but of course resulted in a large performance drop when evaluating on the QuestionBank. The lexicalized parser came out ahead in this experiment,[4] confirming our hypothesis that the latent variable model is better able to pick up the small amount of relevant evidence that is present in the WSJ training data (rather than being systematically

[3] The difference between our shift-reduce parser and the MaltParser are due to small differences in the feature sets.

[4] The F_1 scores were 52.40% vs. 56.39% respectively.

better suited for modeling questions).

3.2 Some Labeled Target Domain Data

In the above experiments, we considered a situation where we have no labeled training data from the target domain, as will typically be the case. We now consider a situation where a small amount of labeled data (2,000 manually parsed sentences) from the domain of interest is available for training.

We experimented with two different ways of utilizing this additional training data. In a first experiment, we trained models on the concatenation of the WSJ and QuestionBank training sets (we did not attempt to weight the different corpora). As Table 2 shows (left columns), even a modest amount of labeled data from the target domain can significantly boost parsing performance, giving double-digit improvements in some cases. While not shown in the table, the parsing accuracies on the WSJ development set where largely unaffected by the additional training data.

Alternatively, one can also train models exclusively on the QuestionBank data, resulting in question specific models. The parsing accuracies of these domain-specific models are shown in the right columns of Table 2, and are significantly lower than those of models trained on the concatenated training sets. They are often times even lower than the results of parsers trained exclusively on the WSJ, indicating that 2,000 sentences are not sufficient to train accurate parsers, even for quite narrow domains.

4 Uptraining for Domain-Adaptation

The results in the previous section suggest that parsers without global constraints have difficulties dealing with the syntactic differences between declarative sentences and questions. A possible explanation is that similar word configurations can appear in both types of sentences, but with very different syntactic interpretation. Local models without global constraints are therefore mislead into dead-end interpretations from which they cannot recover (McDonald and Nivre, 2007). Our approach will therefore be to use a large amount of unlabeled data to bias the model towards the appropriate distribution for the target domain. Rather than looking for feature correspondences between the domains (Blitzer et al., 2006), we propose to use automatically labeled target domain data to learn the target domain distribution directly.

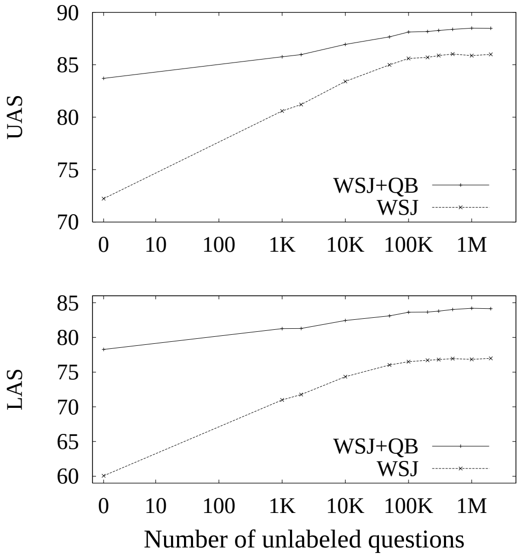


Figure 2: Uptraining with large amounts of unlabeled data gives significant improvements over two different supervised baselines.

4.1 Uptraining vs. Self-training

The idea of training parsers on their own output has been around for as long as there have been statistical parsers, but typically does not work well at all (Charniak, 1997). Steedman et al. (2003) and Clark et al. (2003) present co-training procedures for parsers and taggers respectively, which are effective when only very little labeled data is available. McClosky et al. (2006a) were the first to improve a state-of-the-art constituency parsing system by utilizing unlabeled data for self-training. In subsequent work, they show that the same idea can be used for domain adaptation if the unlabeled data is chosen accordingly (McClosky et al., 2006b). Sagae and Tsujii (2007) co-train two dependency parsers by adding automatically parsed sentences for which the parsers agree to the training data. Finally, Suzuki et al. (2009) present a very effective semi-supervised approach in which features from multiple generative models estimated on unlabeled data are combined in a discriminative system for structured prediction.

All of these approaches have in common that their ultimate goal is to improve the final performance. Our work differs in that instead of improving the

Uptraining with different base parsers	Using only WSJ data			Using WSJ + QB data		
	UAS	LAS	POS	UAS	LAS	POS
Baseline	72.23	60.06	88.48	83.70	78.27	91.32
Self-training	73.62	61.63	89.60	84.26	79.15	92.09
Uptraining on Petrov et al. (2006)	86.02	76.94	90.75	88.38	84.02	93.63
Uptraining on Petrov (2010)	85.21	76.19	90.74	88.63	84.14	93.53

Table 3: Uptraining substantially improves parsing accuracies, while self-training gives only minor improvements.

performance of the best parser, we want to build a more efficient parser that comes close to the accuracy of the best parser. To do this, we parse the unlabeled data with our most accurate parser and generate noisy, but fairly accurate labels (parse trees) for the unlabeled data. We refer to the parser used for producing the automatic labels as the base parser (unless otherwise noted, we used the latent variable parser of Petrov et al. (2006) as our base parser). Because the most accurate base parsers are constituency parsers, we need to convert the parse trees to dependencies using the Stanford converter (see Section 2). The automatically parsed sentences are appended to the labeled training data, and the shift-reduce parser (and the part-of-speech tagger) are trained on this new training set. We did not increase the weight of the WSJ training data, but weighted the QuestionBank training data by a factor of ten in the WSJ+QB experiments.

4.2 Varying amounts of unlabeled data

Figure 2 shows the efficacy of uptraining as a function of the size of the unlabeled data. Both labeled (LAS) and unlabeled accuracies (UAS) improve sharply when automatically parsed sentences from the target domain are added to the training data, and level off after 100,000 sentences. Comparing the end-points of the dashed lines (models having access only to labeled data from the WSJ) and the starting points of the solid lines (models that have access to both WSJ and QuestionBank), one can see that roughly the same improvements (from 72% to 86% UAS and from 60% to 77% LAS) can be obtained by having access to 2,000 labeled sentences from the target domain or uptraining with a large amount of unlabeled data from the target domain. The benefits seem to be complementary and can be combined to give the best results. The final accuracy of 88.63 / 84.14 (UAS / LAS) on the question evaluation set is comparable to the in-domain performance on newswire data (88.24 / 84.69).

4.3 Varying the base parser

Table 3 then compares uptraining on the output of different base parsers to pure self-training. In these experiments, the same set of 500,000 questions was parsed by different base parsers. The automatic parses were then added to the labeled training data and the parser was retrained. As the results show, self-training provides only modest improvements of less than 2%, while uptraining gives double-digit improvements in some cases. Interestingly, there seems to be no substantial difference between uptraining on the output of a single latent variable parser (Petrov et al., 2006) and a product of latent variable grammars (Petrov, 2010). It appears that the roughly 1% accuracy difference between the two base parsers is not important for uptraining.

4.4 POS-less parsing

Our uptraining procedure improves parse quality on out-of-domain data to the level of in-domain accuracy. However, looking closer at Table 3, one can see that the POS accuracy is still relatively low (93.53%), potentially limiting the final accuracy.

To remove this limitation (and also the dependence on a separate POS tagger), we experimented with word cluster features. As shown in Koo et al. (2008), word cluster features can be used in conjunction with POS tags to improve parsing accuracy. Here, we use them instead of POS tags in order to further reduce the domain-dependence of our model. Similar to Koo et al. (2008), we use the Brown clustering algorithm (Brown et al., 1992) to produce a deterministic hierarchical clustering of our input vocabulary. We then extract features based on vary-

	UAS	LAS	POS
Part-of-Speech Tags	88.35	84.05	93.53
Word Cluster Features	87.92	83.73	—

Table 4: Parsing accuracies of uptrained parsers with and without part-of-speech tags and word cluster features.

ing cluster granularities (6 and 10 bits in our experiments). Table 4 shows that roughly the same level of accuracy can be achieved with cluster based features instead of POS tag features. This change makes our parser completely deterministic and enables us to process sentences in a single left-to-right pass.

5 Error Analysis

To provide a better understanding of the challenges involved in parsing questions, we analyzed the errors made by our WSJ-trained shift-reduce parser and also compared them to the errors that are left after uptraining.

5.1 POS errors

Many parsing errors can be traced back to POS tagging errors, which are much more frequent on out-of-domain data than on in-domain data (88.8% on the question data compared to above 95.0% on WSJ data). Part of the reason for the lower POS tagging accuracy is the higher unknown word ratio (7.3% on the question evaluation set, compared to 3.4% on the WSJ evaluation set). Another reason is a change in the lexical distribution.

For example, wh-determiners (WDT) are quite rare in the WSJ training data (relative frequency 0.45%), but five times more common in the QuestionBank training data (2.49%). In addition to this frequency difference, 52.43% of the WDTs in the WSJ are the word “which” and 46.97% are“that”. In the QuestionBank on the other hand, “what” is by far the most common WDT word (81.40%), while “which” and “that” account only for 13.65% and 4.94% respectively. Not surprisingly the most common POS error involves wh-determiners (typically the word “what”) being incorrectly labeled as Wh-pronouns (WP), resulting in head and label errors like the one shown in Figure 3(a).

To separate out POS tagging errors from parsing errors, we also ran experiments with correct (gold) POS tags. The parsing accuracies of our shift-reduce parser using gold POS tags are listed in the last rows of Tables 1 and 2. Even with gold POS tags, the deterministic shift-reduce parser falls short of the accuracies of the constituency parsers (with automatic tags), presumably because the shift-reduce model is making only local decisions and is lacking the global constraints provided by the context-free grammar.

Dep. Label	Frequency	WSJ	Uptrained
nsubj	934	41.02	88.64
amod	556	78.21	86.00
dobj	555	70.10	83.12
attr	471	8.64	93.49
aux	467	77.31	82.56

Table 5: F_1 scores for the most frequent labels in the QuestionBank development set. Uptraining leads to huge improvements compared to training only on the WSJ.

5.2 Dependency errors

To find the main error types, we looked at the most frequent labels in the QuestionBank development set, and analyzed the ones that benefited the most from uptraining. Table 5 has the frequency and F-scores of the dependency types that we are going to discuss in the following. We also provide examples which are illustrated in Figure 3.

nsubj: The WSJ-trained model is often producing parses that are missing a subject (nsubj). Questions like “What is the oldest profession?” and “When was Ozzy Osbourne born?” should have “profession” and “Osbourne” as nsubjs, but in both cases the WSJ-trained parser did not label any subj (see Figures 3(b) and 3(c)). Another common error is to mislabel nsubj. For example, the nsubj of “What are liver enzymes?” should be enzymes, but the WSJ-trained parser labels “What” as the nsubj, which makes sense in a statement but not in a question.

amod: The model is overpredicting “amod”, resulting in low precision figures for this label. An example is “How many points make up a perfect fivepin bowling score?”. The Stanford dependency uses “How” as the head of “many” in noun phrases like “How many points”, and the relation is a generic “dep”. But in the WSJ model prediction, “many’s” head is “points,” and the relation mislabeled as amod. Since it’s an adjective preceding the noun,

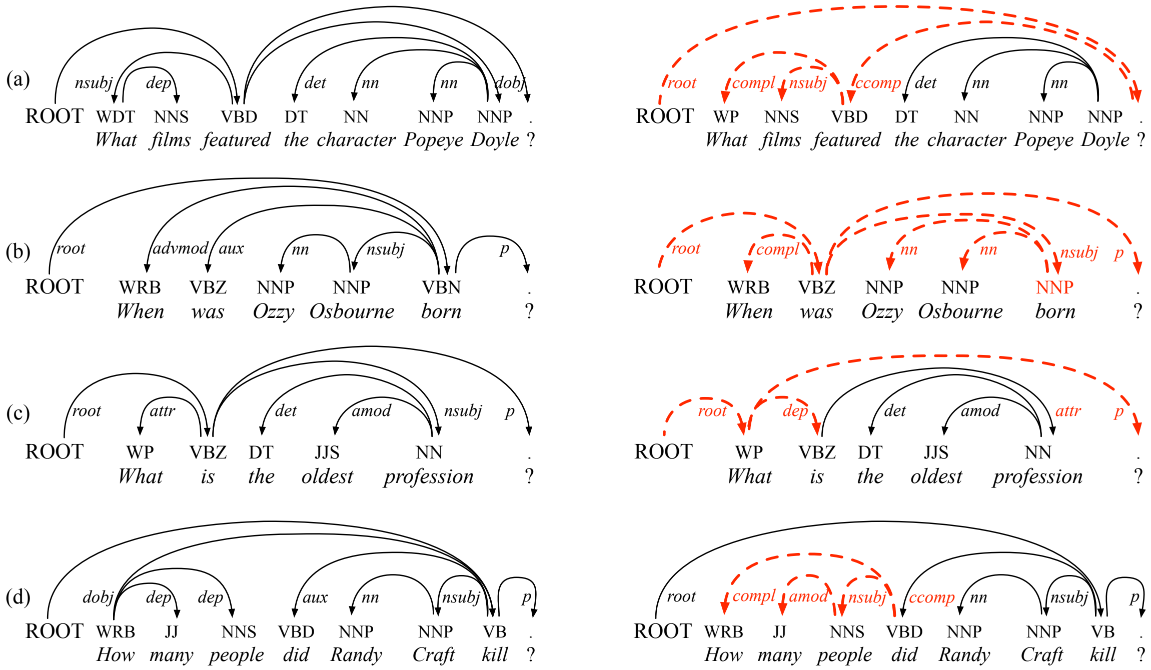

(a)
nsubj dep dobj det nn nn
ROOT WDT NNS VBD DT NN NNP NNP .
What films featured the character Popeye Doyle ?
root compl nsubj ccomp det nn nn
ROOT WP NNS VBD DT NN NNP NNP .
What films featured the character Popeye Doyle ?
(b)
root advmod aux nn nsubj p
ROOT WRB VBZ NNP NNP VBN .
When was Ozzy Osbourne born ?
root compl nn nn nsubj p
ROOT WRB VBZ NNP NNP NNP .
When was Ozzy Osbourne born ?
(c)
root attr det amod nsubj p
ROOT WP VBZ DT JJS NN .
What is the oldest profession ?
root dep det amod attr p
ROOT WP VBZ DT JJS NN .
What is the oldest profession ?
(d)
dobj dep dep aux nn nsubj p
ROOT WRB JJ NNS VBD NNP NNP VB .
How many people did Randy Craft kill ?
root compl amod nsubj ccomp nn nsubj p
ROOT WRB JJ NNS VBD NNP NNP VB .
How many people did Randy Craft kill ?

Figure 3: Example questions from the QuestionBank development set and their correct parses (left), as well as the predictions of a model trained on the WSJ (right).

the WSJ model often makes this mistake and therefore the precision is much lower when it doesn't see more questions in the training data.

dobj: The WSJ model doesn't predict object extraction well. For example, in "How many people did Randy Craft kill?" (Figure 3(d)), the direct object of kill should be "How many people." In the Stanford dependencies, the correct labels for this noun phrase are "dobj dep dep," but the WSJ model predicts "compl amod nsubj." This is a common error caused by the different word order in questions. The uptrained model is much better at handling these type of constructions.

attr: An attr (attributive) is a wh-noun phrase (WHNP) complement of a copular verb. In the WSJ training data, only 4,641 out of 950,028 dependencies are attr (0.5%); in the QuestionBank training data, 1,023 out of 17,069 (6.0%) are attr. As a consequence, the WSJ model cannot predict this label in questions very well.

aux: "What does the abbreviation AIDS stand for?" should have "stand" as the main head of the sentence, and "does" as its aux. However, the WSJ model labeled "does" as the main head. Similar patterns occur in many questions, and therefore the WSJ has a very low recall rate.

In contrast, mostly local labels (that are not related to question/statement structure differences) have a consistently high accuracy. For example: det has an accuracy of 98.86% with the WSJ-trained model, and 99.24% with the uptrained model.

6 Conclusions

We presented a method for domain adaptation of deterministic shift-reduce parsers. We evaluated multiple state-of-the-art parsers on a question corpus and showed that parsing accuracies degrade substantially on this out-of-domain task. Most notably, deterministic shift-reduce parsers have difficulty dealing with the modified word order and lose more than 20% in accuracy. We then proposed a simple, yet very effective *uptraining* method for domain-adaptation. In a nutshell, we trained a deterministic shift-reduce parser on the output of a more accurate, but slower parser. Uptraining with large amounts of unlabeled data gives similar improvements as having access to 2,000 labeled sentences from the target domain. With 2,000 labeled questions and a large amount of unlabeled questions, uptraining is able to close the gap between in-domain and out-of-domain accuracy.

Acknowledgements

We would like to thank Ryan McDonald for running the MstParser experiments and for many fruitful discussions on this topic. We would also like to thank Joakim Nivre for help with the MatlParser and Marie-Catherine de Marneffe for help with the Stanford Dependency Converter.

References

J. Blitzer, R. McDonald, and F. Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP '06*.

A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT '98*.

T. Brants. 2000. TnT – a statistical part-of-speech tagger. In *ANLP '00*.

P. Brown, V. Della Pietra, P. deSouza, J. Lai, and R. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*.

X. Carreras, M. Collins, and T. Koo. 2008. TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *CoNLL '08*.

E. Charniak and M. Johnson. 2005. Coarse-to-Fine N-Best Parsing and MaxEnt Discriminative Reranking. In *ACL'05*.

E. Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *AI '97*.

E. Charniak. 2000. A maximum–entropy–inspired parser. In *NAACL '00*.

S. Clark, J. Curran, and M. Osborne. 2003. Bootstrapping pos-taggers using unlabelled data. In *CoNLL '03*.

M.-C. de Marneffe, B. MacCartney, and C. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC '06*.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *COLING '96*.

J. Foster. 2010. "cba to check the spelling": Investigating parser performance on discussion forum posts. In *NAACL '10*.

D. Gildea. 2001. Corpus variation and parser performance. In *EMNLP '01*.

Z. Huang and M. Harper. 2009. Self-training PCFG grammars with latent annotations across languages. In *EMNLP '09*.

J. Judge, A. Cahill, and J. v. Genabith. 2006. Questionbank: creating a corpus of parse-annotated questions. In *ACL '06*.

T. Koo and M. Collins. 2010. Efficient third-order dependency parsers. In *ACL '10*.

T. Koo, X. Carreras, and M. Collins. 2008. Simple semi-supervised dependency parsing. In *ACL '08*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. In *Computational Linguistics*.

A.F.T. Martins, N.A. Smith, and E.P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *ACL '09*.

D. McClosky, E. Charniak, and M. Johnson. 2006a. Effective self-training for parsing. In *NAACL '06*.

D. McClosky, E. Charniak, and M. Johnson. 2006b. Reranking and self-training for parser adaptation. In *ACL '06*.

R. McDonald and J. Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *EMNLP '07*.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *ACL '05*.

R. McDonald, K. Lerman, and F. Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *CoNLL '06*.

J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kbler, S. Marinov, and E. Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2).

J. Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4).

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL '06*.

S. Petrov. 2010. Products of random latent variable grammars. In *NAACL '10*.

S. Riedel and J. Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *EMNLP '06*.

K. Sagae and A. Lavie. 2006. Parser combination by reparsing. In *NAACL '06*.

K. Sagae and J. Tsujii. 2007. Dependency parsing and domain adaptation with lr models and parser ensembles. In *CoNLL '07*.

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 *EACL '03*.

J. Suzuki, H. Isozaki, X. Carreras, and M. Collins. 2009. An empirical study of semi-supervised structured conditional models for dependency parsing. In *EMNLP '09*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *IWPT '03*.

A Unified Framework for Scope Learning via Simplified Shallow Semantic Parsing

Qiaoming Zhu Junhui Li Hongling Wang Guodong Zhou[*]
School of Computer Science and Technology
Soochow University, Suzhou, China 215006
{qmzhu, lijunhui, hlwang, gdzhou}@suda.edu.cn

Abstract

This paper approaches the scope learning problem via simplified shallow semantic parsing. This is done by regarding the cue as the predicate and mapping its scope into several constituents as the arguments of the cue. Evaluation on the BioScope corpus shows that the structural information plays a critical role in capturing the relationship between a cue and its dominated arguments. It also shows that our parsing approach significantly outperforms the state-of-the-art chunking ones. Although our parsing approach is only evaluated on negation and speculation scope learning here, it is portable to other kinds of scope learning.

1 Introduction

Recent years have witnessed an increasing interest in the analysis of linguistic scope in natural language. The task of scope learning deals with the syntactic analysis of what part of a given sentence is under user's special interest. For example, of negation assertion concerned, a negation cue (e.g., *not*, *no*) usually dominates a fragment of the given sentence, rather than the whole sentence, especially when the sentence is long. Generally, scope learning involves two subtasks: cue recognition and its scope identification. The former decides whether a word or phrase in a sentence is a cue of a special interest, where the semantic information of the word or phrase, rather than the syntactic information, plays a critical role. The latter determines the sequences of words in the sentence which are dominated by the given cue.

Recognizing the scope of a special interest (e.g., negative assertion and speculative assertion) is essential in information extraction (IE), whose aim is to derive factual knowledge from free text. For example, Vincze et al. (2008) pointed out that the extracted information within the scope of a negation or speculation cue should either be discarded or presented separately from factual information. This is especially important in the biomedical and scientific domains, where various linguistic forms are used extensively to express impressions, hypothesized explanations of experimental results or negative findings. Besides, Vincez et al. (2008) reported that 13.45% and 17.70% of the sentences in the abstracts subcorpus of the BioScope corpus contain negative and speculative assertions, respectively, while 12.70% and 19.44% of the sentences in the full papers subcorpus contain negative and speculative assertions, respectively. In addition to the IE tasks in the biomedical domain, negation scope learning has attracted increasing attention in some natural language processing (NLP) tasks, such as sentiment classification (Turney, 2002). For example, in the sentence "*The chair is not comfortable but cheap*", although both the polarities of the words "*comfortable*" and "*cheap*" are positive, the polarity of "the chair" regarding the attribute "*cheap*" keeps positive while the polarity of "the chair" regarding the attribute "*comfortable*" is reversed due to the negation cue "*not*". Similarly, seeing the increasing interest in speculation scope learning, the CoNLL'2010 shared task (Farkas et al., 2010) aims to detect uncertain information in resolving the scopes of speculation cues.

Most of the initial research in this literature focused on either recognizing negated terms or identifying speculative sentences, using some heuristic

[*] Corresponding author

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 714–724,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

rules (Chapman et al., 2001; Light et al., 2004), and machine learning methods (Goldin and Chapman, 2003; Medlock and Briscoe, 2007). However, scope learning has been largely ignored until the recent release of the BioScope corpus (Szarvas et al., 2008), where negation/speculation cues and their scopes are annotated explicitly. Morante et al. (2008) and Morante and Daelemans (2009a & 2009b) pioneered the research on scope learning by formulating it as a chunking problem, which classifies the words of a sentence as being inside or outside the scope of a cue. Alternatively, Özgür and Radev (2009) and Øvrelid et al. (2010) defined heuristic rules for speculation scope learning from constituency and dependency parse tree perspectives, respectively.

Although the chunking approach has been evaluated on negation and speculation scope learning and can be easily ported to other scope learning tasks, it ignores syntactic information and suffers from low performance. Alternatively, even if the rule-based methods may be effective for a special scope learning task (e.g., speculation scope learning), it is not readily adoptable to other scope learning tasks (e.g., negation scope learning). Instead, this paper explores scope learning from parse tree perspective and formulates it as a simplified shallow semantic parsing problem, which has been extensively studied in the past few years (Carreras and Màrquez, 2005). In particular, the cue is recast as the predicate and the scope is recast as the arguments of the cue. The motivation behind is that the structured syntactic information plays a critical role in scope learning and should be paid much more attention, as indicated by previous studies in shallow semantic parsing (Gildea and Palmer, 2002; Punyakanok et al., 2005). Although our approach is evaluated only on negation and speculation scope learning here, it is portable to other kinds of scope learning.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 introduces the Bioscope corpus on which our approach is evaluated. Section 4 describes our parsing approach by formulating scope learning as a simplified shallow semantic parsing problem. Section 5 presents the experimental results. Finally, Section 6 concludes the work.

2 Related Work

Most of the previous research on scope learning falls into negation scope learning and speculation scope learning.

Negation Scope Learning

Morante et al. (2008) pioneered the research on negation scope learning, largely due to the availability of a large-scale annotated corpus, the Bioscope corpus. They approached negation cue recognition as a classification problem and formulated negation scope identification as a chunking problem which predicts whether a word in the sentence is inside or outside of the negation scope, with proper post-processing to ensure consecutiveness of the negation scope. Morante and Daelemans (2009a) further improved the performance by combing several classifiers and achieved the accuracy of ~98% for negation cue recognition and the PCS (Percentage of Correct Scope) of ~74% for negation scope identification on the abstracts subcorpus. However, this chunking approach suffers from low performance, in particular on long sentences. For example, given golden negation cues on the Bioscope corpus, Morante and Daelemans (2009a) only got the performance of 50.26% in PCS on the full papers subcorpus (22.8 words per sentence on average), compared to 87.27% in PCS on the clinical reports subcorpus (6.6 words per sentence on average).

Speculation Scope Learning

Similar to Morante and Daelemans (2009a), Morante and Daelemans (2009b) formulated speculation scope identification as a chunking problem which predicts whether a word in the sentence is inside or outside of the speculation scope, with proper post-processing to ensure consecutiveness of the speculation scope. They concluded that their method for negation scope identification is portable to speculation scope identification. However, of speculation scope identification concerned, it also suffers from low performance, with only 60.59% in PCS for the clinical reports subcorpus of short sentences.

Alternatively, Özgür and Radev (2009) employed some heuristic rules from constituency parse tree perspective on speculation scope identification. Given golden speculation cues, their rule-based method achieves the accuracies of 79.89%

and 61.13% on the abstracts and the full papers subcorpora, respectively. The more recent CoNLL'2010 shared task was dedicated to the detection of speculation cues and their linguistic scope in natural language processing (Farkas et al., 2010). As a representative, Øvrelid et al. (2010) adopted some heuristic rules from dependency parse tree perspective to identify their speculation scopes.

3 Cues and Scopes in the BioScope Corpus

This paper employs the BioScope corpus (Szarvas et al., 2008; Vincze et al., 2008) [1], a freely downloadable resource from the biomedical domain, as the benchmark corpus. In this corpus, every sentence is annotated with negation cues and speculation cues (if it has), as well as their linguistic scopes. Figure 1 shows a self-explainable example. It is possible that a negation/speculation cue consists of multiple words, i.e., "can not"/"*indicate that*" in Figure 1.

```
<sentence id="S26.8">These findings <xcope
id="X26.8.2"><cue type="speculation"
ref="X26.8.2">indicate that</cue> <xcope
id="X26.8.1">corticosteroid resistance in bron-
chial asthma <cue type="negation"
ref="X26.8.1">can not</cue> be explained by
abnormalities in corticosteroid receptor charac-
teristics</xcope></xcope>.</sentence>
```

Figure 1: An annotated sentence in the BioScope corpus

The Bioscope corpus consists of three subcorpora: biological full papers from FlyBase and from BMC Bioinformatics, biological paper abstracts from the GENIA corpus (Collier et al., 1999), and clinical (radiology) reports. Among them, the full papers subcorpus and the abstracts subcorpus come from the same genre, and thus share some common characteristics in statistics, such as the number of words in the negation/speculation scope to the right (or left) of the negation/speculation cue and the average scope length. In comparison, the clinical reports subcorpus consists of clinical radiology reports with short sentences. For detailed statistics and annotation guidelines about the three subcorpora, please see Morante and Daelemans (2009a & 2009b).

For preprocessing, all the sentences in the Bioscope corpus are tokenized and then parsed using the Berkeley parser (Petrov and Klein, 2007) [2] trained on the GENIA TreeBank (GTB) 1.0 (Tateisi et al., 2005)[3], which is a bracketed corpus in (almost) PTB style. 10-fold cross-validation on GTB1.0 shows that the parser achieves the performance of 86.57 in F1-measure. It is worth noting that the GTB1.0 corpus includes all the sentences in the abstracts subcorpus of the Bioscope corpus.

4 Scope Learning via Simplified Shallow Semantic Parsing

In this section, we first formulate the scope learning task as a simplified shallow semantic parsing problem. Then, we deal with it using a simplified shallow semantic parsing framework.

4.1 Formulating Scope Learning as a Simplified Shallow Semantic Parsing Problem

Given a parse tree and a predicate in it, shallow semantic parsing recognizes and maps all the constituents in the sentence into their corresponding semantic arguments (roles) of the predicate or not. As far as scope learning considered, the cue can be regarded as the predicate[4], while its scope can be mapped into several constituents dominated by the cue and thus can be regarded as the arguments of the cue. In particular, given a cue and its scope which covers $word_m$, …, $word_n$, we adopt the following two heuristic rules to map the scope of the cue into several constituents which can be deemed as its arguments in the given parse tree.

1) The cue itself and all of its ancestral constituents are non-arguments.
2) If constituent *X* is an argument of the given cue, then *X* should be the highest constituent dominated by the scope of $word_m$, …, $word_n$. That is to say, *X*'s parent constituent must cross-bracket or include the scope of $word_m$, …, $word_n$.

[1] http://www.inf.u-szeged.hu/rgai/bioscope

[2] http://code.google.com/p/berkeleyparser/

[3] http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA

[4] If a speculation cue consists of multiply words (e.g., *whether or not*), the first word (e.g., *whether*) is chosen to represent the speculation signal. However, the last word (e.g., *not*) is chosen to represent the negation cue if it consists of multiple words (e.g., *can not*).

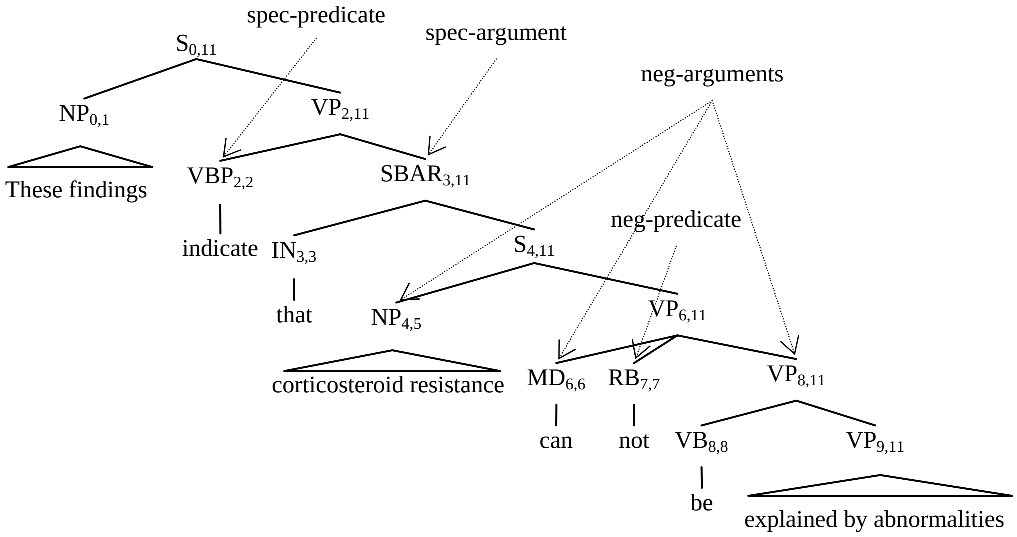


Figure 2: Examples of a negation/speculation cue and its arguments in a parse tree

The first rule ensures that no argument covers the cue while the second rule ensures no overlap between any two arguments. These two constraints between a cue and its arguments are consistent with shallow semantic parsing (Carreras and Màrquez, 2005). For example, in the sentence "*These findings indicate that corticosteroid resistance can not be explained by abnormalities*", the negation cue "*can not*" has the negation scope "*corticosteroid resistance can not be explained by abnormalities*" while the speculation cue "*indicate that*" has the speculation scope "*indicate that corticosteroid resistance can not be explained by abnormalities*". As shown in Figure 2, the node "$RB_{7,7}$" (i.e., *not*) represents the negation cue "*can not*" while its arguments include three constituents {$NP_{4,5}$, $MD_{6,6}$, and $VP_{8,11}$}. Similarly, the node "$VBP_{2,2}$" (i.e., *indicate*) represents the speculation cue "*indicate that*" while its arguments include one constituent $SBAR_{3,11}$. It is worth noting that according to the above rules, scope learning via shallow semantic parsing, i.e. determining the arguments of a given cue, is robust to some variations in the parse trees. This is also empirically justified by our later experiments. For example, if the $VP_{6,11}$ in Figure 2 is incorrectly expanded by the rule $VP_{6,11} \rightarrow MD_{6,6}+RB_{7,7}+VB_{8,8}+VP_{9,11}$, the negation scope of the negation cue "*can not*" can still be correctly detected as long as {$NP_{4,5}$, $MD_{6,6}$, $VB_{8,8}$, and $VP_{9,11}$} are predicated as the arguments of the negation cue "*can not*".

Compared with common shallow semantic parsing which needs to assign an argument with a semantic label, scope identification does not involve semantic label classification and thus could be divided into three consequent phases: argument pruning, argument identification and post-processing.

4.2 Argument Pruning

Similar to the predicate-argument structures in common shallow semantic parsing, the cue-scope structures in scope learning can be also classified into several certain types and argument pruning can be done by employing several heuristic rules accordingly to filter out constituents, which are most likely non-arguments of a given cue. Similar to the heuristic algorithm proposed in Xue and Palmer (2004) for argument pruning in common shallow semantic parsing, the argument pruning algorithm adopted here starts from designating the cue node as the current node and collects its siblings. It then iteratively moves one level up to the parent of the current node and collects its siblings. The algorithm ends when it reaches the root of the parse tree. To sum up, except the cue node itself and its ancestral constituents, any constituent in the parse tree whose parent covers the given cue will be collected as argument candidates. Taking the negation cue node "$RB_{7,7}$" in Figure 2 as an example, constituents {$MD_{6,6}$, $VP_{8,11}$, $NP_{4,5}$, $IN_{3,3}$, VBP$_{2,2}$, and NP$_{0,1}$} are collected as its argument candidates consequently.

Feature	Remarks
B1	Cue itself: the word of the cue, e.g., not, rather_than. (*can_not*)
B2	Phrase Type: the syntactic category of the argument candidate. (*NP*)
B3	Path: the syntactic path from the argument candidate to the cue. (*NP<S>VP>RB*)
B4	Position: the positional relationship of the argument candidate with the cue. “left” or “right”. (*left*)

Table 1: Basic features and their instantiations for argument identification in scope learning, with NP4,5 as the focus constituent (i.e., the argument candidate) and “can not” as the given cue, regarding Figure 2.

Feature	Remarks
Argument Candidate (AC) related	
AC1	The headword (AC1H) and its POS (AC1P). (*resistance*, *NN*)
AC2	The left word (AC2W) and its POS (AC2P). (*that*, *IN*)
AC3	The right word (AC3W) and its POS (AC3P). (*can*, *MD*)
AC4	The phrase type of its left sibling (AC4L) and its right sibling (AC4R). (*NULL*, *VP*)
AC5	The phrase type of its parent node. (*S*)
AC6	The subcategory. (*S:NP+VP*)
Cue/Predicate (CP) related	
CP1	Its POS. (*RB*)
CP2	Its left word (CP2L) and right word (CP2R). (*can*, *be*)
CP3	The subcategory. (*VP:MD+RB+VP*)
CP4	The phrase type of its parent node. (*VP*)
Combined Features related with the Argument Candidate (CFAC1-CFAC2)	
b2&AC1H, b2&AC1P	
Combined Features related with the given Cue/Predicate (CFCP1-CFCP2)	
B1&CP2L, B1&CP2R	
Combined Features related with both the Argument Candidate and the given Cue/Predicate (CFACCP1-CFACCP7)	
B1&B2, B1&B3, B1&CP1, B3&CFCP1, B3&CFCP2, B4&CFCP1, B4&CFCP2	

Table 2: Additional features and their instantiations for argument identification in scope identification, with NP4,5 as the focus constituent (i.e., the argument candidate) and “can not” as the given cue, regarding Figure 2.

4.3 Argument Identification

Here, a binary classifier is applied to determine the argument candidates as either valid arguments or non-arguments. Similar to argument identification in common shallow semantic parsing, the structured syntactic information plays a critical role in scope learning.

Basic Features

Table 1 lists the basic features for argument identification. These features are also widely used in common shallow semantic parsing for both verbal and nominal predicates (Xue, 2008; Li et al., 2009).

Additional Features

To capture more useful information in the cue-scope structures, we also explore various kinds of additional features. Table 2 shows the features in better capturing the details regarding the argument candidate and the cue. In particular, we categorize the additional features into three groups according to their relationship with the argument candidate (AC, in short) and the given cue/predicate (CP, in short).

Some features proposed above may not be effective in argument identification. Therefore, we adopt the greedy feature selection algorithm as described in Jiang and Ng (2006) to pick up positive features incrementally according to their contributions on the development data. The algorithm repeatedly selects one feature each time, which contributes most, and stops when adding any of the remaining features fails to improve the performance.

4.4 Post-Processing

Although a cue in the BioScope corpus always has only one continuous block as its scope (including the cue itself), the scope identifier may result in discontinuous scope due to independent predication in the argument identification phase. Given the golden negation/speculation cues, we observe that 6.2%/9.1% of the negation/speculation scopes predicted by our scope identifier are discontinuous.

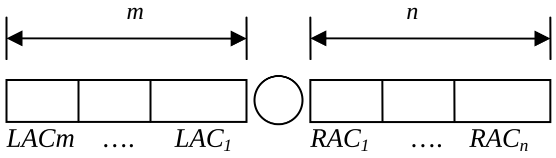


Figure 3: Projecting the left and the right argument candidates into the word level.

Figure 3 demonstrates the projection of all the argument candidates into the word level. According to our argument pruning algorithm in Section 4.2, except the words presented by the cue, the projection covers the whole sentence and each constituent (*LAC*$_i$ or *RAC*$_j$ in Figure 3) receives a probability distribution of being an argument of the given cue in the argument identification phase.

Since a cue is deemed inside its scope in the BioScope corpus, our post-processing algorithm first includes the cue in its scope and then starts to identify the left and the right scope boundaries, respectively.

As shown in Figure 3, the left boundary has *m+1* possibilities, namely the cue itself, the leftmost word of constituent *LAC*$_i$ (1<=*i*<=*m*). Supposing *LAC*$_i$ receives probability of P_i being an argument, we use the following formula to determine *LAC*$_{k*}$ whose leftmost word represents the boundary of the left scope. If k^*=0, then the cue itself represents its left boundary.

$$k^* = \arg\max_k \prod_{i=1}^{k} P_i * \prod_{i=k+1}^{m} (1 - P_i)$$

Similarly, the right boundary of the given cue can be decided.

4.5 Cue Recognition

Automatic recognition of cues of a special interest is the prerequisite for a scope learning system. The approaches to recognizing cues of a special interest usually fall into two categories: 1) substring matching approaches, which require a set of cue words or phrases in advance (e.g., Light et al., 2004); 2) machine learning approaches, which train a classifier with either supervised or semi-supervised learning methods (e.g., Özgür and Radev, 2009; Szarvas, 2008). Without loss of generality, we adopt a machine learning approach and train a classifier with supervised learning. In particular, we make an independent classification for each word with a BIO label to indicate whether it is the first word of a cue, inside a cue, or outside of it, respectively.

Inspired by previous studies on similar tasks such as WSD and nominal predicate recognition in shallow semantic parsing (Lee and Ng, 2002; Li et al., 2009), where various features on the word itself, surrounding words and syntactic information have been successfully used, we believe that such information is also valuable to automatic recognition of cues. Table 3 shows the features employed for cue recognition. In particular, we categorize these features into three groups: 1) features about the cue candidate itself (CC in short); 2) features about surrounding words (SW in short); and 3) structural features derived from the syntactic parse tree (SF in short).

Feature	**Remarks**
Cue Candidate (CC) related	
CC1	The cue candidate itself. (*indicate*)
CC2	The stem of the cue candidate. (*indicate*)
CC3	The POS tag of the cue candidate. (*VBP*)
Surrounding Words (SW) related	
SW1	The left surrounding words with the window size of 3. (*these*, *findings*)
SW2	The right surrounding words with the window size of 3. (*that*, *corticosteroid*, *resistance*)
Structural Features (SF)	
SF1	The subcategory of the candidate node. (*VP-->VBP+SBAR*)
SF2	The subcategory of the candidate node's parent. (*S-->NP+VP*)
SF3	POS tag of the candidate node + Phrase type of its parent node + Phrase type of its grandpa node. (*VBP* + *VP* + *S*)

Table 3: Features and their instantiations for cue recognition, with VBP$_{2,2}$ as the cue candidate, regarding Figure 2.

5 Experimentation

We have evaluated our simplified shallow semantic parsing approach to negation and speculation scope learning on the BioScope corpus.

5.1 Experimental Settings

Following the experimental setting in Morante et al. (2008) and Morante and Daelemans (2009a & 2009b), the abstracts subcorpus is randomly divided into 10 folds so as to perform 10-fold cross-validation, while the performance on both the papers and clinical reports subcorpora is evaluated using the system trained on the whole abstracts subcorpus. In addition, SVMLight[5] is selected as our classifier.

For cue recognition, we report its performance using precision/recall/F1-measure. For scope identification, we report the accuracy in PCS (Percentage of Correct Scopes) when the golden cues are given, and report precision/recall/F1-measure when the cues are automatically recognized.

5.2 Experimental Results on Golden Parse Trees and Golden Cues

In order to select beneficial features from the additional features proposed in Section 4.3, we randomly split the abstracts subcorpus into the training data and the development data with proportion of 4:1. After performing the greedy feature selection algorithm on the development data, 7 features {CFACCP5, CP2R, CFCP1, AC1P, CP3, CFACCP7, AC4R} are selected consecutively for negation scope identification while 11 features {CFACCP5, AC2W, CFACCP2, CFACCP4, AC5, CFCP1, CFACCP7, CFACCP1, CP4, AC3P, CFAC2} are selected for speculation scope identification. Table 4 gives the contribution of additional features on the development data. It shows that the additional features significantly improve the performance by 11.66% in accuracy from 74.93% to 86.59% ($\chi^2; p < 0.01$) for negation scope identification and improve the performance by 11.07% in accuracy from 77.29% to 88.36% ($\chi^2; p < 0.01$) for speculation scope identification. The feature selection experiments suggest that the features (e.g., CFACCP5, AC2W, CFCP1) related to neighboring words of the cue play a critical role for both negation and speculation scope identification. This may be due to the fact that neighboring words usually imply important sentential information. For example, "*can not be*" indicates a passive clause while "*did not*" indicates an active clause.

Since the additional selected features significantly improve the performance for both negation and speculation scope identification, we will include those additional selected features in all the remaining experiments.

Task	Features	Acc (%)
Negation scope identification	Baseline +selected features	74.93 86.59
Speculation scope identification	Baseline +selected features	77.29 88.36

Table 4: Contribution of additional selected features on the development dataset of the abstracts subcorpus

Since all the sentences in the abstracts subcorpus are included in the GTB1.0 corpus while we do not have golden parse trees for the sentences in the full papers and the clinical reports subcorpora, we only evaluate the performance of scope identification on the abstracts subcorpus with golden parse trees. Table 5 presents the performance on the abstracts subcorpus by performing 10-fold cross-validation. It shows that given golden parse trees and golden cues, speculation scope identification achieves higher performance (e.g., ~3.3% higher in accuracy) than negation scope identification. This is mainly due to the observation on the BioScope corpus that the scope of a speculation cue can be usually characterized by its POS and the syntactic structures of the sentence where it occurs. For example, the scope of a verb in active voice usually starts at the cue itself and ends at its object (e.g., the speculation cue "*indicate that*" in Figure 2 scopes the fragment of "*indicate that corticosteroid resistance can not be explained by abnormalities*"). Moreover, the statistics on the abstracts subcorpus shows that the number of arguments per speculation cue is smaller than that of arguments per negation cue (e.g., 1.5 vs. 1.8).

Task	Acc (%)
Negation scope identification	83.10
Speculation scope identification	86.41

Table 5: Accuracy (%) of scope identification with golden parse trees and golden cues on the abstracts subcorpus using 10-fold cross-validation

It is worth nothing that we adopted the post-processing algorithm proposed in Section 4.4 to ensure the continuousness of identified scope. As to examine the effectiveness of the algorithm, we abandon the proposed algorithm by simply taking the left and right-most boundaries of any nodes in the tree which are classified as in scope. Experiments on the abstracts subcorpus using 10-fold cross-validation shows that the simple post-processing rule gets the performance of 80.59 and 86.08 in accuracy for negation and speculation

[5] http://svmlight.joachims.org/

scope identification, respectively, which is lower than the performance in Table 5 achieved by our post-processing algorithm.

5.3 Experimental Results on Automatic Parse Trees and Golden Cues

The GTB1.0 corpus contains 18,541 sentences in which 11,850 of them (63.91%) overlap with the sentences in the abstracts subcorpus[6]. In order to get automatic parse trees, we train the Berkeley parser with the remaining 6,691 sentences in GTB1.0, which achieves the performance of 85.22 in F1-measure on the remaining 11,850 sentences in GTB1.0. Table 6 shows the performance of scope identification on automatic parse trees and golden cues. In addition, we also report an oracle performance to explore the best possible performance of our system by assuming that our scope finder can always correctly determine whether a candidate is an argument or not. That is, if an argument candidate falls within the golden scope, then it is a argument. This is to measure the impact of automatic syntactic parsing itself. Table 6 shows that:

1) For both negation and speculaiton scope identification, automatic syntactic parsing lowers the performance on the abstracts subcorpus (e.g., from 83.10% to 81.84% in accuracy for negation scope identification and from 86.41% to 83.74% in accuracy for speculaiton scope identification). However, the performance drop shows that both negation and speculation scope identification are not as senstive to automatic syntactic parsing as common shallow semantic parsing, whose performance might decrease by about ~10 in F1-measure (Toutanova et al., 2005). This indicates that scope identification via simplified shallow semantic parsing is robust to some variations in the parse trees.
2) Although speculation scope identification consistently achieves higher performance than negaiton scope identification when golden parse trees are availabe, speculation scope identification achieves comparable performance with negation scope identification on the abstracts subcorpus and the full papers subcorpus while speculation scope identification even performs ~20% lower in accuracy than negation scope identification on the clinical report subcorpus. This is largely due to that specuaiton scope identification is more sensitive to syntactic parsing errors than negation scope identification due to the wider scope of a speculation cue while the sentences of the clinical reports come from a different genre, which indicates low performance in syntactic parsing.
3) Given the performance gap between the performance of our scope finder and the oracle performance, there is still much room for further performance improvement.

Task	Method	Abstracts	Papers	Clinical
Negation scope identification	auto	81.84	62.70	85.21
	oracle	94.37	83.33	98.39
Speculation scope identification	auto	83.74	61.29	67.90
	oracle	95.69	83.72	83.29

Table 6: Accuracy (%) of scope identification on the three subcorpora using automatic parser trained on 6,691 sentences in GTB1.0

Task	Method	Abstracts	Papers	Clinical
Negation scope identification	M et al. (2008)	57.33	n/a	n/a
	M & D (2009a)	73.36	50.26	87.27
	Our baseline	73.42	53.70	88.42
	Our final	81.84	64.02	89.79
Speculation scope identification	M & D (2009b)	77.13	47.94	60.59
	Ö & R (2009)	79.89	61.13	n/a
	Our baseline	77.39	54.55	61.92
	Our final	83.74	63.49	68.78

Table 7: Performance comparison of our system with the state-of-the-art ones in accuracy (%). Note that all the performances achieved on the full papers subcorpus and the clinical subcorpus are achieved using the whole GTB1.0 corpus of 18,541 sentences while all the performances achieved on the abstract subcorpus are achieved using 6,691 sentences from GTB1.0 due to overlap of the abstract subcorpus with GTB1.0.

Table 7 compares our performance with related ones. It shows that even our baseline system with the four basic features presented in Table 1 achieves comparable performance with Morante et al. (2008) and Morante and Daelemans (2009a & 2009b). This further indicates the appropriateness of our simplified shallow semantic parsing approach and the effectiveness of structured syntactic information on scope identification. It also shows that our final system significantly outperforms the

[6] There are a few cases where two sentences in the abstracts subcorpus map into one sentence in GTB1.0.

state-of-the-art ones using a chunking approach, especially on the abstracts and full papers subcorpora. However, the improvement on the clinical reports subcorpora for negation scope identification is much less apparent, partly due to the fact that the sentences in this subcorpus are much simpler (with average length of 6.6 words per sentence) and thus a chunking approach can achieve high performance. Table 7 also shows that our parsing approach to speculation scope identification outperforms the rule-based method in Özgür and Radev (2009), where 10-fold cross-validation is performed on both the abstracts and the full papers subcorpora.

5.4 Experimental Results with Automatic Parse Trees and Automatic Cues

So far negation/speculation cues are assumed to be manually annotated and available. Here we turn to a more realistic scenario in which cues are automatically recognized. In the following, we first report the results of cue recognition and then the results of scope identification with automatic cues.

Cue Recognition

Task	Features	R (%)	P (%)	F1
Negation cue recognition	CC + SW	93.80	94.39	94.09
	CC+SW+SF	95.50	95.72	95.61
Speculation cue recognition	CC + SW	83.77	92.04	87.71
	CC+SW+SF	84.33	93.07	88.49

Table 8: Performance of automatic cue recognition with gold parse trees on the abstracts subcorpus using 10-fold cross-validation

Table 8 lists the performance of cue recognition on the abstracts subcorpus, assuming all words in the sentences as candidates. It shows that as a complement to features derived from word/pos information (CC+SW features), structural features (SF features) derived from the syntactic parse tree significantly improve the performance of cue recognition by about 1.52 and 0.78 in F1-measure for negation and speculation cue recognition, respectively, and thus included thereafter. In addition, we have also experimented on only these words, which happen to be a cue or inside a cue in the training data as cue candidates. However, this experimental setting achieves a lower performance than that when all words are considered.

Task	Corpus	R (%)	P (%)	F1
Negation cue recognition	Abstracts	94.99	94.35	94.67
	Papers	90.48	87.47	88.95
	Clinical	86.81	88.54	87.67
Speculation cue recognition	Abstracts	83.74	93.14	88.19
	Papers	73.02	82.31	77.39
	Clinical	33.33	91.77	48.90

Table 9: Performance of automatic cue recognition with automatic parse trees on the three subcorpora

Table 9 presents the performance of cue recognition achieved with automatic parse trees on the three subcorpora. It shows that:

1) The performance gap of cue recognition between golden parse trees and automatic parse trees on the abstracts subcorpus is not salient (e.g., 95.61 vs. 94.67 in F1-measure for negation cues and 88.49 vs. 88.19 for speculation cues), largely due to the features defined for cue recognition are local and insenstive to syntactic variations.
2) The performance of negation cue recognition is higher than that of speculation cue recognition on all the three subcorpora. This is prabably due to the fact that the collection of negation cue words or phrases is limitted while speculation cue words or phrases are more open. This is illustrated by our statistics that about only 1% and 1% of negation cues in the full papers and the clinical reports subcorpora are absent from the abstracts subcorpus, compared to about 6% and 20% for speculation cues.
3) Unexpected, the recall of speculation cue recognition on the clinical reports subcorpus is very low (i.e., 33.33% in recall measure). This is probably due to the absence of about 20% speculation cues from the training data of the abstracts subcorpus. Moreover, the speculation cue "*or*", which accounts for about 24% of specuaiton cues in the clinical reports subcorpus, only acheives about 2% in recall largely due to the errors caused by the classifier trained on the abstracts subcorpus, where only about 11% of words "*or*" are annotated as speculation cues.

Scope Identification with Automatic Cue Recognition

Table 10 lists the performance of both negation and speculation scope identification with automatic cues and automatic parse trees. It shows that automatic cue recognition lowers the performance by

3.34, 6.80, and 8.38 in F1-measure for negation scope identification on the abstracts, the full papers and the clinical reports subcorpora, respectively, while it lowers the performance by 6.50, 13.14 and 31.23 in F1-measures for speculation scope identification on the three subcorpora, respectively, suggesting the big challenge of cue recognition in the two scope learning tasks.

Task	Corpus	R (%)	P (%)	F1
Negation scope identification	Abstracts	78.77	78.24	78.50
	Papers	58.20	56.27	57.22
	Clinical	80.62	82.22	81.41
Speculation scope identification	Abstracts	73.34	81.58	77.24
	Papers	47.51	53.55	50.35
	Clinical	25.59	70.46	37.55

Table 10: Performance of both negation and speculation scope identification with automatic cues and automatic parse trees

6 Conclusion

In this paper we have presented a new approach to scope learning by formulating it as a simplified shallow semantic parsing problem, which has been extensively studied in the past few years. In particular, we regard the cue as the predicate and map its scope into several constituents which are deemed as arguments of the cue. Evaluation on the Bioscope corpus shows the appropriateness of our parsing approach and that structured syntactic information plays a critical role in capturing the domination relationship between a cue and its dominated arguments. It also shows that our parsing approach outperforms the state-of-the-art chunking ones. Although our approach is only evaluated on negation and speculation scope learning here, it is portable to other kinds of scope learning.

For the future work, we will explore tree kernel-based methods to further improve the performance of scope learning in better capturing the structural information, and apply our parsing approach to other kinds of scope learning.

Acknowledgments

This research was supported by Projects 60873150, 60970056, and 90920004 under the National Natural Science Foundation of China, Project 20093201110006 under the Specialized Research Fund for the Doctoral Program of Higher Education of China.

References

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. *CoNLL' 2005*.

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*, 34: 301-310.

Nigel Collier, Hyun Seok Park, Norihiro Ogata, et al. 1999. The GENIA Project: Corpus-Based Knowledge Acquisition and Information Extraction from Genome Research Papers. *EACL'1999*.

Richárd Farkas, Veronika Vincze, György Móra, János Csirik, and György Szarvas. 2010. The CoNLL-2010 Shared Task: Learning to Detect Hedges and their Scope in Natural Language Text. *CoNLL'2010: Shared Task*.

Daniel Gildea and Martha Palmer. 2002. The Necessity of Parsing for Predicate Argument Recognition. *ACL'2002*.

Ilya M. Goldin and Wendy W. Chapman. 2003. Learning to Detect Negation with 'Not' in Medical Texts. *SIGIR'2003*.

Zheng Ping Jiang and Hwee Tou Ng. 2006. Semantic Role Labeling of NomBank: A Maximum Entropy Approach. *EMNLP' 2006*.

Yoong Keok Lee and Hwee Tou Ng. 2002. An Empirical Evaluation of Knowledge Sources and Learning Algorithms for Word Sense Disambiguation. *EMNLP'2002*.

Junhui Li, Guodong Zhou, Hai Zhao, Qiaoming Zhu, and Peide Qian. 2009. Improving Nominal SRL in Chinese Language with Verbal SRL Information and Automatic Predicate Recognition. *EMNLP' 2009*.

Marc Light, Xin Ying Qiu, and Padmini Srinivasan. 2004. The Language of Bioscience: Facts, Speculations, and Statements in Between. *BioLink'2004*.

Ben Medlock and Ted Briscoe. 2007. Weakly Supervised Learning for Hedge Classification in Scientific Literature. *ACL'2007*.

Roser Morante, Anthony Liekens, and Walter Daelemans. 2008. Learning the Scope of Negation in Biomedical Texts. *EMNLP'2008*.

Roser Morante and Walter Daelemans. 2009a. A Metalearning Approach to Processing the Scope of Negation. *CoNLL'2009*.

Roser Morante and Walter Daelemans. 2009b. Learning the Scope of Hedge Cues in Biomedical Texts. *BioNLP'2009*.

Lilja Øvrelid, Erik Velldal, and Stephan Oepen. 2010. Syntactic Scope Resolution in Uncertainty Analysis. *COLING'2010*.

Arzucan Özgür, Dragomir R. Radev. 2009. Detecting Speculations and their Scopes in Scientific Text. *EMNLP'2009*.

Slav Petrov and Dan Klein. 2007. Improved Inference for Unlexicalized Parsing. *NAACL'2007*.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2005. The Necessity of Syntactic Parsing for Semantic Role Labeling. *IJCAI' 2005*.

György Szarvas. 2008. Hedge Classification in Biomedical Texts with a Weakly Supervised Selection of Keywords. *ACL'2008*.

György Szarvas, Veronika Vincze, Richárd Farkas, and János Csirik. 2008. The BioScope corpus: Annotation for Negation, Uncertainty and their Scope in Biomedical Texts. *BioNLP'2008*.

Yuka Tateisi, Akane Yakushiji, Tomoko Ohta, and Jun'ichi Tsujii. 2005. Syntax Annotation for the GENIA Corpus. *IJCNLP'2005* (Companion volume).

Peter D. Turney. 2002. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. *ACL'2002*.

Veronika Vincze, György Szarvas, Richárd Farkas, György Móra, and János Csirik. 2008. The BioScope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC Bioinformatics*, 9(Suppl 11):S9.

Nianwen Xue and Martha Palmer. 2004. Calibrating Features for Semantic Role Labeling. *EMNLP'2004*.

Nianwen Xue. 2008. Labeling Chinese Predicates with Semantic Roles. *Computational Linguistics*, 34(2):225-255.

A New Approach to Lexical Disambiguation of Arabic Text

Rushin Shah
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
rnshah@cs.cmu.edu

Paramveer S. Dhillon, Mark Liberman, Dean Foster, Mohamed Maamouri and Lyle Ungar
University of Pennsylvania
3451 Walnut Street
Philadelphia, PA 19104, USA
{dhillon|myl|ungar}@cis.upenn.edu,
foster@wharton.upenn.edu,
maamouri@ldc.upenn.edu

Abstract

We describe a model for the lexical analysis of Arabic text, using the lists of alternatives supplied by a broad-coverage morphological analyzer, SAMA, which include stable lemma IDs that correspond to combinations of broad word sense categories and POS tags. We break down each of the hundreds of thousands of possible lexical labels into its constituent elements, including lemma ID and part-of-speech. Features are computed for each lexical token based on its local and document-level context and used in a novel, simple, and highly efficient two-stage supervised machine learning algorithm that overcomes the extreme sparsity of label distribution in the training data. The resulting system achieves accuracy of 90.6% for its first choice, and 96.2% for its top two choices, in selecting among the alternatives provided by the SAMA lexical analyzer. We have successfully used this system in applications such as an online reading helper for intermediate learners of the Arabic language, and a tool for improving the productivity of Arabic Treebank annotators.

1 Background and Motivation

This paper presents a methodology for generating high quality lexical analysis of highly inflected languages, and demonstrates excellent performance applying our approach to Arabic. Lexical analysis of the written form of a language involves resolving, explicitly or implicitly, several different kinds of ambiguities. Unfortunately, the usual ways of talking about this process are also ambiguous, and our general approach to the problem, though not unprecedented, has uncommon aspects. Therefore, in order to avoid confusion, we begin by describing how we define the problem.

In an inflected language with an alphabetic writing system, a central issue is how to interpret strings of characters as forms of words. For example, the English letter-string 'winds' will normally be interpreted in one of four different ways, all four of which involve the sequence of two formatives ***wind+s***. The stem 'wind' might be analyzed as (1) a noun meaning something like "air in motion", pronounced [w**I**nd] , which we can associate with an arbitrary but stable identifier like ***wind_n1***; (2) a verb ***wind_v1*** derived from that noun, and pronounced the same way; (3) a verb ***wind_v2*** meaning something like "(cause to) twist", pronounced [wa**I**nd]; or (4) a noun ***wind_n2*** derived from that verb, and pronounced the same way. Each of these "lemmas", or dictionary entries, will have several distinguishable senses, which we may also wish to associate with stable identifiers. The affix '-s' might be analyzed as the plural inflection, if the stem is a noun; or as the third-person singular inflection, if the stem is a verb.

We see this analysis as conceptually divided into four parts: 1) *Morphological analysis*, which recognizes that the letter-string 'winds' might be (perhaps among other things) ***wind/N*** + ***s/PLURAL*** or ***wind/V*** + ***s/3SING***; 2) *Morphological disambiguation*, which involves deciding, for example, that in the phrase "the four winds", 'winds' is probably a plural noun, i.e. ***wind/N*** + ***s/PLURAL***; 3) *Lemma analysis*, which involves recognizing that the stem wind in 'winds' might be any of the four lemmas listed above – perhaps with a further listing of senses or other sub-entries for each of them; and 4) *Lemma disambiguation*, deciding, for example, that

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 725–735,
MIT, Massachusetts, USA, 9-11 October 2010.

the phrase "the four winds" probably involves the lemma ***wind_n1***.

Confusingly, the standard word-analysis tasks in computational linguistics involve various combinations of pieces of these logically-distinguished operations. Thus, "part of speech (POS) tagging" is mainly what we've called "morphological disambiguation", except that it doesn't necessarily require identifying the specific stems and affixes involved. In some cases, it also may require a small amount of "lemma disambiguation", for example to distinguish a proper noun from a common noun. "Sense disambiguation" is basically a form of what we've called "lemma disambiguation", except that the sense disambiguation task may assume that the part of speech is known, and may break down lexical identity more finely than our system happens to do. "Lemmatization" generally refers to a radically simplified form of "lemma analysis" and "lemma disambiguation", where the goal is simply to collapse different inflected forms of any similarly-spelled stems, so that the strings 'wind', 'winds', 'winded', 'winding' will all be treated as instances of the same thing, without in fact making any attempt to determine the identity of "lemmas" in the traditional sense of dictionary entries.

Linguists use the term *morphology* to include all aspects of lexical analysis under discussion here. But in most computational applications, "morphological analysis" does not include the disambiguation of lemmas, because most morphological analyzers do not reference a set of stable lemma IDs. So for the purposes of this paper, we will continue to discuss lemma analysis and disambiguation as conceptually distinct from morphological analysis and disambiguation, although, in fact, our system disambiguates both of these aspects of lexical analysis at the same time.

The lexical analysis of textual character-strings is a more complex and consequential problem in Arabic than it is in English, for several reasons. First, Arabic inflectional morphology is more complex than English inflectional morphology is. Where an English verb has five basic forms, for example, an Arabic verb in principle may have dozens. Second, the Arabic orthographic system writes elements such as prepositions, articles, and possessive pronouns without setting them off by spaces, roughly as if the English phrase "in a way" were written "inaway". This leads to an enormous increase in the number of distinct "orthographic words", and a substantial increase in ambiguity. Third, short vowels are normally omitted in Arabic text, roughly as if English "in a way" were written "nway".

As a result, a whitespace/punctuation-delimited letter-string in Arabic text typically has many more alternative analyses than a comparable English letter-string does, and these analyses have many more parts, drawn from a much larger vocabulary of form-classes. While an English "tagger" can specify the morphosyntactic status of a word by choosing from a few dozen tags, an equivalent level of detail in Arabic would require thousands of alternatives. Similarly, the number of lemmas that might play a role in a given letter-sequence is generally much larger in Arabic than in English.

We start our labeling of Arabic text with the alternative analyses provided by SAMA v. 3.1, the Standard Arabic Morphological Analyzer (Maamouri et al., 2009). SAMA is an updated version of the earlier Buckwalter analyzers (Buckwalter, 2004), with a number of significant differences in analysis to make it compatible with the LDC Arabic Treebank 3-v3.2 (Maamouri et al., 2004). The input to SAMA is an Arabic orthographic word (a string of letters delimited by whitespace or punctuation), and the output of SAMA is a set of alternative analyses, as shown in Table 1. For a typical word, SAMA produces approximately a dozen alternative analyses, but for certain highly ambiguous words it can produce hundreds of alternatives.

The SAMA analyzer has good coverage; for typical texts, the correct analysis of an orthographic word can be found somewhere in SAMA's list of alternatives about **95%** of the time. However, this broad coverage comes at a cost; the list of analytic alternatives must include a long Zipfian tail of rare or contextually-implausible analyses, which collectively are correct often enough to make a large contribution to the coverage statistics. Furthermore, SAMA's long lists of alternative analyses are not evaluated or ordered in terms of overall or contextual plausibility. This makes the results less useful in most practical applications.

Our goal is to rank these alternative analyses so that the correct answer is as near to the top of the list

Token	Lemma	Vocalization	Segmentation	Morphology	Gloss
yHlm	Halam-u_1	yaHolumu	ya + Holum + u	IV3MS + IV + IV-SUFF_MOOD:I	he / it + dream + [ind.]
yHlm	Halam-u_1	yaHoluma	ya + Holum + a	IV3MS + IV + IV-SUFF_MOOD:S	he / it + dream + [sub.]
yHlm	Halum-u_1	yaHolumo	ya + Holum + o	IV3MS + IV + IV-SUFF_MOOD:J	he / it + be gentle + [jus.]
qbl	qabil-a_1	qabila	qabil + a	PV + PV-SUFF_SUBJ:3MS	accept/receive/approve + he/it [verb]
qbl	qabol_1	qabol	qabol	NOUN	Before

Table 1: Partial output of SAMA for ***yHlm*** and ***qbl***. On average, every token produces more than 10 such analyses

as possible. Despite some risk of confusion, we'll refer to SAMA's list of alternative analyses for an orthographic word as potential *labels* for that word. And despite a greater risk of confusion, we'll refer to the assignment of probabilities to the set of SAMA labels for a particular Arabic word in a particular textual context as *tagging*, by analogy to the operation of a stochastic part-of-speech tagger, which similarly assigns probabilities to the set of labels available for a word in textual context.

Although our algorithms have been developed for the particular case of Arabic and the particular set of lexical-analysis labels produced by SAMA, they should be applicable without modification to the sets of labels produced by any broad-coverage lexical analyzer for the orthographic words of any highly-inflected language.

In choosing our approach, we have been motivated by two specific applications. One application aims to help learners of Arabic in reading text, by offering a choice of English glosses with associated Arabic morphological analyses and vocalizations. SAMA's excellent coverage is an important basis for this help; but SAMA's long, unranked list of alternative analyses for a particular letter-string, where many analyses may involve rare words or alternatives that are completely implausible in the context, will be confusing at best for a learner. It is much more helpful for the list to be ranked so that the correct answer is almost always near the top, and is usually one of the top two or three alternatives.

In our second application, this same sort of ranking is also helpful for the linguistically expert native speakers who do Arabic Treebank analysis. These annotators understand the text without difficulty, but find it time-consuming and fatiguing to scan a long list of rare or contextually-implausible alternatives for the correct SAMA output. Their work is faster and more accurate if they start with a list that is ranked accurately in order of contextual plausibility.

Other applications are also possible, such as vocalization of Arabic text for text-to-speech synthesis, or lexical analysis for Arabic parsing. However, our initial goals have been to rank the list of SAMA outputs for human users.

We note in passing that the existence of set of stable "lemma IDs" is an unusual feature of SAMA, which in our opinion ought to be emulated by approaches to lexical analysis in other languages. The lack of such stable lemma IDs has helped to disguise the fact that without lemma analysis and disambiguation, morphological analyses and disambiguation is only a partial solution to the problem of lexical analysis.

In principle, it is obvious that lemma disambiguation and morphological disambiguation are mutually beneficial. If we know the answer to one of the questions, the other one is easier to answer. However, these two tasks require rather different sets of contextual features. Lemma disambiguation is similar to the problem of word-sense disambiguation – on some definitions, they are identical – and as a result, it benefits from paragraph-level and document-level bag-of-words attributes that help to characterize what the text is "about" and therefore which lemmas are more likely to play a role in it. In contrast, morphological disambiguation mainly depends on features of nearby words, which help to characterize how inflected forms of these lemmas might fit into local phrasal structures.

2 Problem and Methodology

Consider a collection of tokens (observations), t_i, referred to by index $i \in \{1, \ldots, n\}$, where each token is associated with a set of p features, x_{ij}, for the j^{th} feature, and a label, l_i, which is a combination of a lemma and a morphological analysis. We use indicator functions y_{ik} to indicate whether or not the k^{th} label for the i^{th} token is present. We represent the complete set of features and labels for the entire training data using matrix notation as X and Y, respectively. Our goal is to predict the label l (or equivalently, the vector y for a given feature vector x.

A standard linear regression model of this problem would be

$$\mathbf{y} = \mathbf{x}\beta + \epsilon \tag{1}$$

The standard linear regression estimate of β (ignoring, for simplicity the fact that the ys are 0/1) is:

$$\hat{\beta} = (X_{train}^T X_{train})^{-1} X_{train}^T Y_{train} \tag{2}$$

where Y_{train} is an $n \times h$ matrix containing 0s and 1s indicating whether or not each of the h possible labels is the correct label (l_i) for each of the n tokens t_i, X_{train} is an $n \times p$ matrix of context features for each of the n tokens, the coefficients $\hat{\beta}$ are $p \times h$.

However, this is a large, sparse, multiple label problem, and the above formulation is neither statistically nor computationally efficient. Each observation $(\mathbf{x}, \mathbf{y})$ consists of thousands of features associated with thousands of potential labels, almost all of which are zero. Worse, the matrix of coefficients β, to be estimated is large ($p \times h$) and one should thus use some sort of transfer learning to share strength across the different labels.

We present a novel principled and highly computationally efficient method of estimating this multilabel model. We use a two stage procedure, first using a subset (X_{train1}, Y_{train1}) of training data to give a fast approximate estimate of β; we then use a second smaller subset of the training data $(X_{train2}, Y_{train2},)$ to "correct" these estimates in a way that we will show can be viewed as a specialized shrinkage. Our first stage estimation approximates β, but avoids the expensive computation of $(X_{train}^T X_{train})^{-1}$. Our second stage corrects (shrinks) these initial estimates in a manner specialized to this problem. The second stage takes advantage of the fact that we only need to consider those candidate labels produced by SAMA. Thus, only dozens of the thousands of possible labels are considered for each token.

We now present our algorithm. We start with a corpus D of documents d of labeled Arabic text. As described above, each token, t_i is associated with a set of features characterizing its context, computed from the other words in the same document, and a label, $l_i = (\text{lemma}_i, \text{morphology}_i)$, which is a combination of a lemma and a morphological analysis. As described below, we introduce a novel factorization of the morphology into 15 different components.

Our estimation algorithm, shown in Algorithm 1, has two stages. We partition the training corpus into two subsets, one of which (X_{train1}) is used to estimate the coefficients βs and the other of which (X_{train2}) is used to optimally "shrink" these coefficient estimates to reduce variance and prevent overfitting due to data sparsity.

For the first stage of our estimation procedure, we simplify the estimate of the (β) matrix (Equation 2) to avoid the inversion of the very high dimensional ($p \times p$) matrix ($X^T X$) by approximating ($X^T X$) by its diagonal, Var(X), the inverse of which is trivial to compute; i.e. we estimate β using

$$\hat{\beta} = \text{Var}(X_{train1})^{-1} X_{train1}^T Y_{train1} \tag{3}$$

For the second stage, we assume that the coefficients for each feature can be shrunk differently, but that coefficients for each feature should be shrunk the same regardless of what label they are predicting. Thus, for a given observation we predict:

$$\hat{g}_{ik} = \sum_{j=1}^{p} w_j \hat{\beta}_{jk} x_{ij} \tag{4}$$

where the weights w_j indicate how much to shrink each of the p features.

In practice, we fold the variance of each of the j features into the weight, giving a slightly modified equation:

$$\hat{g}_{ik} = \sum_{j=1}^{p} \alpha_j \beta^*_{jk} x_{ij} \tag{5}$$

where $\beta^* = X_{train1}^T Y_{train1}$ is just a matrix of the counts of how often each context feature shows up with each label in the first training set. The vector α, which we will estimate by regression, is just the shrinkage weights w rescaled by the feature variance.

Note that the formation here is different from the first stage. Instead of having each observation be a token, we now let each observation be a (token, label) pair, but only include those labels that were output by SAMA. For a given token t_i and potential label l_k, our goal is to approximate the indicator function $g(i, k)$, which is 1 if the k^{th} label of token t_i is present, and 0 otherwise. We find candidate labels using a morphological analyzer (namely SAMA), which returns a set of possible candidate labels, say $C(t)$, for each Arabic token t. Our predicted label for t_i is then $\text{argmax}_{k \in C(t_i)} g(i, k)$.

The regression model for learning the weights α_j in the second stage thus has a row for each label $g(i, k)$ associated with a SAMA candidate for each token $i = n_{train1+1} \ldots n_{train2}$ in the second training set. The value of $g(i, k)$ is predicted as a function of the feature vector $z_{ijk} = \beta_{jk}^* x_{ij}$.

The shrinkage coefficients, α_j, could be estimated from theory, using a version of James-Stein shrinkage (James and Stein, 1961), but in practice, superior results are obtained by estimating them empirically. Since there are only p of them (unlike the $p * h$ βs), a relatively small training set is sufficient. We found that regression-SVMs work slightly better than linear regression and significantly better than standard classification SVMs for this problem.

Prediction is then done in the obvious way by taking the tokens in a test corpus D_{test}, generating context features and candidate SAMA labels for each token t_i, and selected the candidate label with the highest score $\hat{g}(i, k)$ that we set out to learn. More formally, The model parameters β^* and α produced by the algorithm allow one to estimate the most likely label for a new token t_i out of a set of candidate labels $C(t_i)$ using

$$k_{pred} = \text{argmax}_{k \in C(t_i)} \sum_{j=1}^{p} \alpha_j \beta_{jk}^* x_{ij} \qquad (6)$$

The most expensive part of the procedure is estimating β^*, which requires for each token in corpus D_1, (a subset of D), finding the co-occurrence frequencies of each label element (a lemma, or a part of the morphological segmentation) with the target token and jointly with the token and with other tokens or characters in the context of the token of interest. For example, given an Arabic token, ***"yHlm"***, we count what fraction of the time it is associated with each lemma (e.g. ***Halam-u_1***), *count(lemma=**Halam-u_1**, token=**yHlm**)* and each segment (e.g. "ya"), *count(segment=**ya**, token=**yHlm**)*. (Of course, most tokens never show up with most lemmas or segments; this is not a problem.) We also find the base rates of the components of the labels (e.g., *count(lemma=**Halam-u_1**)*, and what fraction of the time the label shows up in various contexts, e.g. *count(lemma=**Halam-u_1**, previous token = **yHlm**)*. We describe these features in more detail below.

Algorithm 1 Training algorithm.

```
Input: A training corpus $D_{train}$ of $n$ observations $(X_{train}, Y_{train})$
Partition $D_{train}$ into two sets, $D_1$ and $D_2$, of sizes $n_{train1}$ and $n_{train2} = n - n_{train1}$ observations
// Using $D_1$, estimate $\beta^*$
$\beta_{jk}^* = \sum_{i=1}^{n_{train1}} x_{ij} y_{ik}$ for the $j^{th}$ feature and $k^{th}$ label
// Using $D_2$, estimate $\alpha_j$
// Generate new "features" $Z$ and the true labels $g(i, k)$ for each of the SAMA candidate labels for each of the tokens in $D_2$
$z_{ijk} = \beta_{jk}^* x_{ij}$ for $i$ in $i = n_{train1} + 1 \ldots n_{train2}$
Estimate $\alpha_j$ for the above (feature,label) pairs $(z_{ijk}, g(i, k))$ using Regression SVMs
Output: $\alpha$ and $\beta^*$
```

3 Features and Labels used for Training

Our approach to tagging Arabic differs from conventional approaches in the two-part shrinkage-based method used, and in the choice of both features and labels used in our model. For features, we study both local context variables, as described above, and document-level word frequencies. For the labels, the key question is what labels are included and how they are factored. Standard "taggers" work by doing an n-way classification of all the alternatives, which is not feasible here due to the thousands of possible labels. Standard approaches such as Conditional Random Fields (CRFs) are intractable with so many labels. Moreover, few if any taggers do any lemma disambiguation; that is partly because one must start with some standard inventory of lemmas, which are not available for most languages, perhaps because the importance of lemma disambiguation has been underestimated.

We make a couple of innovations to deal with these issues. First, we perform lemma disambiguation in addition to “tagging”. As mentioned above, lemmas and morphological information are not independent; the choice of lemma often influences morphology and vice versa. For example, Table 1 contains two analyses for the word **qbl**. For the first analysis, where the lemma is ***qabil-a_1*** and the gloss is *accept/receive/approve + he/it [verb]*, the word is a verb. However, for the second analysis, where the lemma is ***qabol_1*** and the gloss is *before*, the word is a noun.

Simultaneous lemma disambiguation and tagging introduces additional complexity: An analysis of ATB and SAMA shows that there are approximately 2,200 possible morphological analyses (“tags”) and 40,000 possible lemmas; even accounting for the fact that most combinations of lemmas and morphological analyses don’t occur, the size of the label space is still in the order of tens of thousands. To deal with data sparsity, our second innovation is to factor the labels. We factor each label l into a set of 16 label elements (**LE**s). These include lemmas, as well as morphological elements such as basic part-of-speech, suffix, gender, number, mood, etc. These are explained in detail below. Thus, since each label l is a set of 15 categorical variables, each $\mathbf{y}$ in the first learning stage is actually a vector with 16 nonzero components and thousands of zeros. Since we do simultaneous estimation of the entire set of label elements, the value $g(i, k)$ being predicted in the second learning phase is 1 if the entire label set is correct, and zero otherwise. We do *not* learn separate models for each label.

3.1 Label Elements (LEs)

The fact that there are tens of thousands of possible labels presents the problem of extreme sparsity of label distribution in the training data. We find that a model that estimates coefficients β^* to predict a single label (a label being in the Cartesian product of the set of label elements) yields poor performance. Therefore, as just mentioned, we factor each label l into a set of label elements (**LE**s), and learn the correlations β^* between features and label elements, rather than features and entire label sets. This reduces, but does not come close to eliminating, the problem sparsity. A complete list of these **LE**s and their possible values is detailed in Table 2.

LE	**Description**
lemma	Lemma
pre1	Closer prefix
pre2	Farther prefix
det	Determiner
pos	Basic POS
dpos	Additional data on basic pos
suf	Suffix
perpos	Person (basic pos)
numpos	Number (basic pos)
genpos	Gender (basic pos)
persuf	Person (suffix)
numsuf	Number (suffix)
gensuf	Gender (suffix)
mood	Mood of verb
pron	Pronoun suffix

Table 2: Label Elements (**LE**s). Examples of additional data on basic POS include whether a noun is proper or common, whether a verb is transitive or not, etc. Both the basic POS and its suffix may have person, gender and number data.

3.2 Features

3.2.1 Local Context Features

We take (t, l) pairs from D_2, and for each such pair generate features Z based on co-occurrence statistics β^* in D_1, as mentioned in Algorithm 2. These statistics include unigram co-occurrence frequencies of each label with the target token and bigram co-occurrence of the label with the token and with other tokens or characters in the context of the target token. We define them formally in Table 3. Let $Z_{baseline}$ denote the set of all such basic features based on the local context statistics of the target token, namely the words and letters preceding and following it. We will use this set to create a baseline model.

Statistic	Description
Freq	$count_{D_1}(t, l)$
PrevWord	$count_{D_1}(t, l, t_{-1})$
NextWord	$count_{D_1}(t, l, t_{+1})$
PreviLetter	$count_{D_1}(t, l, \text{first letter}(t_{-1}))$
NextiLetter	$count_{D_1}(t, l, \text{first letter}(t_{+1})$
PrevfLetter	$count_{D_1}(t, l, \text{last letter}(t_{-1})$
NextfLetter	$count_{D_1}(t, l, \text{last letter}(t_{+1})$

Table 3: Co-occurrence statistics β^*. We use these to generate feature sets for our regression SVMs.

For each label element (**LE**) e, we define a set of features Z_e similar to $Z_{baseline}$; these features are based on co-occurrence frequencies of the particular **LE** e, not the entire label l.

Finally, we define an aggregate feature set Z_{aggr} as follows:

$$Z_{aggr} = Z_{baseline} \quad \bigcup \quad \{Z_e\} \qquad (7)$$

where $e \in \{$*lemma, pre1, pre2, det, pos, dpos, suf, perpos, numpos, genpos, persuf, numsuf, gensuf, mood, pron*$\}$.

3.2.2 Document Level Features

When trying to predict the lemma, it is useful to include not just the words and characters immediately adjacent to the target token, but also the all the words in the document. These words capture the "topic" of the document, and help to disambiguate different lemmas, which tend to be used or not used based on the topic being discussed, similarly to the way that word sense disambiguation systems in English sometimes use the "bag of words" the document to disambiguate, for example a "bank" for depositing money from a "bank" of a river. More precisely, we augment the features for each target token with the counts of each word in the document (the "term frequency" *tf*) in which the token occurs with a given label.

$$Z_{full} = Z_{aggr} \quad \bigcup \quad Z_{tf} \qquad (8)$$

This set Z_{full} is our final feature set. We use Z_{full} to train an SVM model M_{full}; this is our final predictive model.

3.3 Corpora used for Training and Testing

We use three modules of the Penn Arabic Treebank (ATB) (Maamouri et al., 2004), namely ATB1, ATB2 and ATB3 as our corpus of labeled Arabic text, D. Each ATB module is a collection of newswire data from a particular agency. ATB1 uses the Associated Press as a source, ATB2 uses Ummah, and ATB3 uses Annahar. D contains a total of 1,835 documents, accounting for approximately 350,000 words. We construct the training and testing sets D_{train} and D_{test} from D using 10-fold cross validation, and we construct D_1 and D_2 from D_{train} by randomly performing a 9:1 split.

As mentioned earlier, we use the SAMA morphological analyzer to obtain candidate labels $C(t)$ for each token t while training and testing an SVM model on D_2 and D_{test} respectively. A sample output of SAMA is shown in Table 1. To improve coverage, we also add to $C(t)$ all the labels l seen for t in D_1. We find that doing so improves coverage to **98%**. This is an upper bound on the accuracy of our model.

$$C(t) = \text{SAMA}(t) \quad \bigcup \quad \{l|(t, l) \in D_1\} \qquad (9)$$

4 Results

We use two metrics of accuracy: **A1**, which measures the percentage of tokens for which the model assigns the highest score to the correct label or **LE** value (or **E1**$= 100 - A1$, the corresponding percentage error), and **A2**, which measures the percentage of tokens for which the correct label or **LE** value is one of the two highest ranked choices returned by the model (or **E2** = $100 - A2$). We test our model M_{full} on D_{test} and achieve **A1** and **A2** scores of 90.6% and 96.2% respectively. The accuracy achieved by our M_{full} model is, to the best of our knowledge, higher than prior approaches have been able to achieve so far for the problem of combined morphological and lemma disambiguation. This is all the more impressive considering that the upper bound on accuracy for our model is **98%** because, as described above, our set of candidate labels is incomplete.

In order to analyze how well different **LE**s can be predicted, we train an SVM model M_e for each **LE** e using the feature set Z_e, and test all such models on D_{test}. The results for all the **LE**s are reported in the form of error percentages **E1** and **E2** in Table 4.

Model	**E1**	**E2**	**Model**	**E1**	**E2**
M_{lemma}	11.1	4.9	M_{pre1}	1.9	1.4
M_{pre2}	0.2	0	M_{det}	0.7	0.1
M_{pos}	23.4	4.0	M_{dpos}	10.3	1.9
M_{suf}	7.6	2.5	M_{perpos}	3.0	0.1
M_{numpos}	3.2	0.2	M_{genpos}	1.8	0.1
M_{persuf}	3.2	0.1	M_{numsuf}	8.2	0.5
M_{gensuf}	11.6	0.4	M_{mood}	1.6	1.4
M_{pron}	1.8	0.6	M_{case}	14.7	5.9
$\boldsymbol{M_{full}}$	**9.4**	**3.8**	-	-	-

Table 4: Results of M_e for each **LE** e. **Note:** The results reported are 10 fold cross validation test accuracies and no parameters have been tuned on them.

A comparison of the results for M_{full} with the results for M_{lemma} and M_{pos} is particularly informative. We see that M_{full} is able to achieve a substantially lower **E1** error score (9.4%) than M_{lemma} (11.1%) and M_{pos} (23.4%); in other words, we find that our full model is able to predict lemmas and basic parts-of-speech more accurately than the individual models for each of these elements.

We examine the effect of varying the size of D_2, i.e. the number of SVM training instances, on the performance of M_{full} on D_{test}, and find that with increasing sizes of D_2, **E1** reduces only slightly from 9.5% to 9.4%, and shows no improvement thereafter. We also find that the use of document-level features in M_{lemma} reduces **E1** and **E2** percentages for M_{lemma} by 5.7% and 3.2% respectively.

4.1 Comparison to Alternate Approaches

4.1.1 Structured Prediction Models

Preliminary experiments showed that knowing the predicted labels (lemma + morphology) of the surrounding words can slightly improve the predictive accuracy of our model. To further investigate this effect, we tried running experiments using different structured models, namely CRF (Conditional Random Fields) (Lafferty et al., 2001), (Structured) MIRA (Margin Infused Relaxation Algorithm) (Crammer et al., 2006) and Structured Perceptron (Collins, 2002). We used linear chain CRFs as implemented in MALLET Toolbox (McCallum, 2001) and for Structured MIRA and Perceptron we used their implementations from EDLIN Toolbox (Ganchev and Georgiev, 2009). However, given the vast label space of our problem, running these methods proved infeasible. The time complexity of these methods scales badly with the number of labels; It took a week to train a linear chain CRF for only $\sim$ 50 labels and though MIRA and Perceptron are online algorithms, they also become intractable beyond a few hundred labels. Since our label space contains combinations of lemmas and morphologies, so even after factoring, the dimension of the label space is in the order of thousands.

We also tried a naïve version (two-pass approximation) of these structured models. In addition to the features in Z_{full}, we include the predicted labels for the tokens preceding and following the target token as features. This new model is not only slow to train, but also achieves only slightly lower error rates (1.2% lower **E1** and 1.0% lower **E2**) than M_{full}. This provides an upper bound on the benefit of using the more complex structured models, and suggests that given their computational demands our (unstructured) model M_{full} is a better choice.

4.1.2 MADA

(Habash and Rambow, 2005) perform morphological disambiguation using a morphological analyzer. (Roth et al., 2008) augment this with lemma disambiguation; they call their system MADA. Our work differs from theirs in a number of respects. Firstly, they don't use the two step regression procedure that we use. Secondly, they use only "unigram" features. Also, they do not learn a single model from a feature set based on labels and **LE**s; instead, they combine models for individual elements by using weighted agreement. We trained and tested MADA v2.32 using its full feature set on the same D_{train} and D_{test}. We should point out that this is not an exact comparison, since MADA uses the older Buckwalter morphological analyzer.[1]

4.1.3 Other Alternatives

Unfactored Labels: To illustrate the benefit obtained by breaking down each label l into

[1] A new version of MADA was released very close to the submission deadline for this conference.

LEs, we contrast the performance of our M_{full} model to an SVM model $M_{baseline}$ trained using only the feature set $Z_{baseline}$, which only contains features based on entire labels, those based on individual **LE**s.

Independent lemma and morphology prediction: Another alternative approach is to predict lemmas and morphological analyses separately. We construct a feature set $Z_{lemma'} = Z_{full} - Z_{lemma}$ and train an SVM model $M_{lemma'}$ using this feature set. Labels are then predicted by simply combining the results predicted independently by M_{lemma} and $M_{lemma'}$. Let M_{ind} denote this approach.

Unigram Features: Finally, we also consider a context-less approach, i.e. using only "unigram" features for labels as well as **LE**s. We call this feature set Z_{uni}, and the corresponding SVM model M_{uni}.

The results of these various models, along with those of M_{full} are summarized in Table 5. We see that M_{full} has roughly half the error rate of the state-of-the-art MADA system.

Model	**E1**	**E2**
$M_{baseline}$	13.6	9.1
M_{ind}	18.7	6.0
M_{uni}	11.6	6.4
M_{cheat}	8.2	2.8
MADA	**16.9**	**12.6**
$\boldsymbol{M_{full}}$	**9.4**	**3.8**

Table 5: Percent error rates of alternative approaches. **Note:** The results reported are 10 fold cross validation test accuracies and no parameters have been tuned on them. We used same train-test splits for all the datasets.

5 Related Work

(Hajic, 2000) show that for highly inflectional languages, the use of a morphological analyzer improves accuracy of disambiguation. (Diab et al., 2004) perform tokenization, POS tagging and base phrase chunking using an SVM based learner. (Ahmed and Nürnberger, 2008) perform word-sense disambiguation using a Naive Bayesian model and rely on parallel corpora and matching schemes instead of a morphological analyzer. (Kulick, 2010) perform simultaneous tokenization and part-of-speech tagging for Arabic by separating closed and open-class items and focusing on the likelihood of possible stems of open-class words. (Mohamed and Kübler, 2010) present a hybrid method between word-based and segment-based POS tagging for Arabic and report good results. (Toutanova and Cherry, 2009) perform joint lemmatization and part-of-speech tagging for English, Bulgarian, Czech and Slovene, but they do not use the two step estimation-shrinkage model described in this paper; nor do they factor labels. The idea of joint lemmatization and part-of-speech tagging has also been discussed in the context of Hungarian in (Kornai, 1994).

A substantial amount of relevant work has been done previously for Hebrew. (Adler and Elhadad, 2006) perform Hebrew morphological disambiguation using an unsupervised morpheme-based HMM, but they report lower scores than those achieved by our model. Moreover, their analysis doesn't include lemma IDs, which is a novelty of our model. (Goldberg et al., 2008) extend the work of (Adler and Elhadad, 2006) by using an EM algorithm, and achieve an accuracy of 88% for full morphological analysis, but again, this does not include lemma IDs. To the best of our knowledge, there is no existing research for Hebrew that does what we did for Arabic, namely to use simultaneous lemma and morphological disambiguation to improve both. (Dinur et al., 2009) show that prepositions and function words can be accurately segmented using unsupervised methods. However, by using this method as a preprocessing step, we would lose the power of a *simultaneous solution* for these problems. Our method is closer in style to a CRF, giving much of the accuracy gains of simultaneous solution, while being about 4 orders of magnitude easier to train.

We believe that our use of factored labels is novel for the problem of simultaneous lemma and morphological disambiguation; however, (Smith et al., 2005) and (Hatori et al., 2008) have previously made use of features based on parts of labels in CRF models for morphological disambiguation and word-sense disambiguation respectively. Also, we note that there is a similarity between our two-stage machine learning approach and log-linear models in machine translation that break the data in two parts, estimating log-probabilities of generative models from one part, and discriminatively re-weighting the models using the second part.

6 Conclusions

We introduced a new approach to accurately predict labels consisting of both lemmas and morphological analyses for Arabic text. We obtained an accuracy of over 90% – substantially higher than current state-of-the-art systems. Key to our success is the factoring of labels into lemma and a large set of morphosyntactic elements, and the use of an algorithm that computes a simple initial estimate of the coefficient relating each contextual feature to each label element (simply by counting co-occurrence) and then regularizes these features by shrinking each of the coefficients for each feature by an amount determined by supervised learning using only the candidate label sets produced by SAMA.

We also showed that using features of word n-grams is preferable to using features of only individual tokens of data. Finally, we showed that a model using a full feature set based on labels as well as factored components of labels, which we call label elements (**LE**s) works better than a model created by combining individual models for each **LE**. We believe that the approach we have used to create our model can be successfully applied not just to Arabic but also to other languages such as Turkish, Hungarian and Finnish that have highly inflectional morphology. The current accuracy of of our model, getting the correct answer among the top two choices 96.2% of the time is high enough to be highly useful for tasks such as aiding the manual annotation of Arabic text; a more complete automation would require that accuracy for the single top choice.

Acknowledgments

We woud like to thank everyone at the Linguistic Data Consortium, especially Christopher Cieri, David Graff, Seth Kulick, Ann Bies, Wajdi Zaghouani and Basma Bouziri for their help. We also wish to thank the anonymous reviewers for their comments and suggestions.

References

Meni Adler and Michael Elhadad. 2006. An Unsupervised Morpheme-Based HMM for Hebrew Morphological Disambiguation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*.

Farag Ahmed and Andreas Nürnberger. 2008. Arabic/English Word Translation Disambiguation using Parallel Corpora and Matching Schemes. In *Proceedings of EAMT'08*, Hamburg, Germany.

Tim Buckwalter. 2004. Buckwalter Arabic Morphological Analyzer version 2.0.

Michael Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of EMNLP'02*.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online Passive-Aggressive Algorithms. *Journal of Machine Learning Research*, 7:551–585.

Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. 2004. Automatic Tagging of Arabic text: From Raw Text to Base Phrase Chunks. In *Proceedings of the 5th Meeting of the North American Chapter of the Association for Computational Linguistics/Human Language Technologies Conference (HLT-NAACL'04)*.

Elad Dinur, Dmitry Davidov, and Ari Rappoport. 2009. Unsupervised Concept Discovery in Hebrew Using Simple Unsupervised Word Prefix Segmentation for Hebrew and Arabic. In *Proceedings of the EACL 2009 Workshop on Computational Approaches to Semitic Languages*.

Kuzman Ganchev and Georgi Georgiev. 2009. Edlin: An Easy to Read Linear Learning Framework. In *Proceedings of RANLP'09*.

Yoav Goldberg, Meni Adler, and Michael Elhadad. 2008. EM Can Find Pretty Good HMM POS-Taggers (When Given a Good Start)*. In *Proceedings of ACL'08*.

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, MI, USA.

Jan Hajic. 2000. Morphological Tagging: Data vs. Dictionaries. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL'00)*.

Jun Hatori, Yusuke Miyao, and Jun'ichi Tsujii. 2008. Word Sense Disambiguation for All Words using Tree-Structured Conditional Random Fields. In *Proceedings of COLing'08*.

W. James and Charles Stein. 1961. Estimation with Quadratic Loss. In *Proceedings of the Fourth Berkeley*

Symposium on Mathematical Statistics and Probability, Volume 1.

András Kornai. 1994. On Hungarian morphology (Linguistica, Series A: Studia et Dissertationes 14). *Linguistics Institute of Hungarian Academy of Sciences, Budapest*.

Seth Kulick. 2010. Simultaneous Tokenization and Part-of-Speech Tagging for Arabic without a Morphological Analyzer. In *Proceedings of ACL'10*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of ICML'01*, pages 282–289.

Mohamed Maamouri, Ann Bies, and Tim Buckwalter. 2004. The Penn Arabic Treebank: Building a Large Scale Annotated Arabic Corpus. In *Proceedings of NEMLAR Conference on Arabic Language Resources and Tools*.

Mohamed Maamouri, David Graff, Basma Bouziri, Sondos Krouna, and Seth Kulick. 2009. LDC Standard Arabic Morphological Analyzer (SAMA) v. 3.0.

Andrew McCallum, 2001. *MALLET: A Machine Learning for Language Toolkit*. Software available at `http://mallet.cs.umass.edu`.

Emad Mohamed and Sandra Kübler. 2010. Arabic Part of Speech Tagging. In *Proceedings of LREC'10*.

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 *Proceedings of ACL'08*, Columbus, Ohio, USA.

Noah A. Smith, David A. Smith, and Roy W. Tromble. 2005. Context-Based Morphological Disambiguation with Random Fields*. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*.

Kristina Toutanova and Colin Cherry. 2009. A Global Model for Joint Lemmatization and Part-of-Speech Prediction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing*, pages 486–494.

What a Parser can Learn from a Semantic Role Labeler and *Vice Versa*

Stephen A. Boxwell, Dennis N. Mehay, Chris Brew
The Ohio State University
{boxwell, mehay, cbrew}@ling.ohio-state.edu

Abstract

In many NLP systems, there is a unidirectional flow of information in which a parser supplies input to a semantic role labeler. In this paper, we build a system that allows information to flow in both directions. We make use of semantic role predictions in choosing a single-best parse. This process relies on an averaged perceptron model to distinguish likely semantic roles from erroneous ones. Our system penalizes parses that give rise to low-scoring semantic roles. To explore the consequences of this we perform two experiments. First, we use a baseline generative model to produce n-best parses, which are then re-ordered by our semantic model. Second, we use a modified version of our semantic role labeler to predict semantic roles at parse time. The performance of this modified labeler is weaker than that of our best full SRL, because it is restricted to features that can be computed directly from the parser's packed chart. For both experiments, the resulting semantic predictions are then used to select parses. Finally, we feed the selected parses produced by each experiment to the full version of our semantic role labeler. We find that SRL performance can be improved over this baseline by selecting parses with likely semantic roles.

1 Introduction

In the semantic role labeling task, words or groups of words are described in terms of their relations to a predicate. For example, the sentence *Robin admires Leslie* has two semantic role-bearing words: *Robin* is the agent or experiencer of the *admire* predicate, and *Leslie* is the patient. These semantic relations are distinct from syntactic relations like subject and object – the proper nouns in the sentence *Leslie is admired by Robin* have the same semantic relationships as *Robin admires Leslie*, even though the syntax differs.

Although syntax and semantics do not always align with each other, they are correlated. Almost all automatic semantic role labeling systems take a syntactic representation of a sentence (taken from an automatic parser or a human annotator), and use the syntactic information to predict semantic roles. When a semantic role labeler predicts an incorrect role, it is often due to an error in the parse tree. Consider the erroneously annotated sentence from the Penn Treebank corpus shown in Figure 1. If a semantic role labeling system relies heavily upon syntactic attachment decisions, then it will likely predict that *in 1956* describes the time that asbestos was used, rather than when it ceased to be used.

Errors of this kind are common in treebanks and in automatic parses. It is telling, though, that while the hand-annotated Penn Treebank (Marcus et al., 1993), the Charniak parser (Charniak, 2001), and the C&C parser (Clark and Curran, 2004) all produce the erroneous parse from Figure 1, the hand-annotated Propbank corpus of verbal semantic roles (Palmer et al., 2005) correctly identifies *in 1956* as a temporal modifier of *stopped*, rather than *using*. This demonstrates that while syntactic attachment decisions like these are difficult for humans and for automatic parsers, a human reader has little difficulty identifying the correct semantic relationship between the temporal modifier and the verbs. This is likely due to the fact that the meaning suggested by the parse in Figure 1 is unlikely – the reader instinctively feels that a temporal modifier fits better with the verb *stop* than with the verb *use*.

In this paper, we will use the idea that semantic roles predicted by correct parses are more natural than semantic roles predicted by erroneous parses. By modifying a state-of-the-art CCG semantic role labeler to predict semantic roles at parse time, or by using it to select from an n-best list, we can prefer analyses that yield likely semantic roles. Syntactic analysis is treated not as an autonomous task, but rather as a contributor to the final goal of semantic role labeling.

2 Related Work

There has been a great deal of work in joint parsing and semantic role labeling in recent years. Two notable efforts have been the CoNLL 2008 and 2009 shared tasks

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 736–744,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

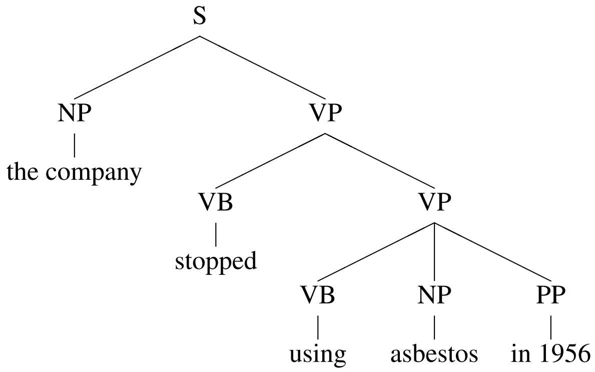


Figure 1: A parse tree based on the treebank parse of wsj_0003.3. Notice that the temporal adjunct is erroneously attached low. In a syntax-based SRL system, this will likely lead to a role prediction error.

(Surdeanu et al., 2008; Hajič et al., 2009). Many of these systems perform joint syntactic and semantic analysis by generating an n-best list of syntactic parses, labeling semantic roles on all of them, then re-ranking these parses by some means. Our approach differs from this strategy by abandoning the preliminary ranking and predicting semantic roles at parse time. By doing this, we effectively open semantic roles in the entire parse forest to examination by the ranking model, rather than restricting the model to an n-best list generated by a baseline parser. The spirit of this work more closely resembles that of Finkel and Manning (2009) , which improves both parsing and named entity recognition by combining the two tasks.

3 Why Predicting Semantic Roles in a Packed Chart is Difficult

Predicting semantic roles in the environment of a packed chart is difficult when using an atomic CFG. In order to achieve the polynomial efficiency appropriate for wide-coverage parsing, it is necessary to "pack" the chart – that is, to combine distinct analyses of a given span of words that produce the same category. The only other widely used option for wide-coverage parsing is to use beam search with a narrow beam, which runs the risk of search errors. On methodological grounds we prefer an exhaustive search, since systems that rely heavily on heuristics for their efficiency are difficult to understand, debug or improve. It is straightforward to read off the highest scoring parse from a packed chart, and similarly routine to generate an n-best list containing a highly-ranked subset of the parses. However, a packed chart built on an atomic CFG does not make available all of the features that are important to many CFG-based SRL systems. In particular, the very useful treepath feature, which lists the categories touched by walking the tree from the predicate to the target word, only makes sense when you have a complete tree, so cannot easily be computed from the chart (Figure 2). Chart edges can be lexicalized with their headwords, and this information would be useful in role labeling – but even this misses vital subcategorization information that would be available in the complete parse. An ideal formalism for our purpose would condense into the category label a wide range of information about combinatory potential, heads, and syntactic dependencies. At the same time it should allow the creation of a packed chart, come with labeled training data, and have a high-quality parser and semantic role labeler already available. Fortunately, Combinatory Categorial Grammar offers these desiderata, so this is our formalism of choice.

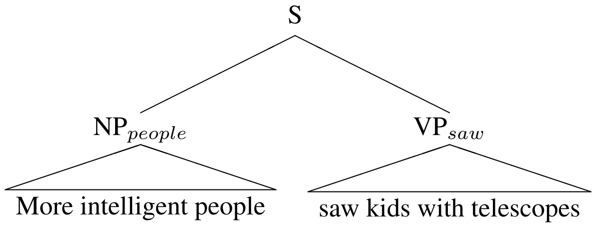


Figure 2: In the context of a packed chart, it is meaningless to speak of a treepath between *saw* and *people* because multiple analyses are "packed" under a single category.

4 Combinatory Categorial Grammar

Combinatory Categorial Grammar (Steedman, 2000) is a grammar formalism that describes words in terms of their combinatory potential. For example, determiners belong to the category np/n, or "the category of words that become noun phrases when combined with a noun to the right". The rightmost category indicates the argument that the category is seeking, the leftmost category indicates the result of combining this category with its argument, and the slash (/ or \) indicates the direction of combination. Categories can be nested within each other: a transitive verb like *devoured* belongs to the category

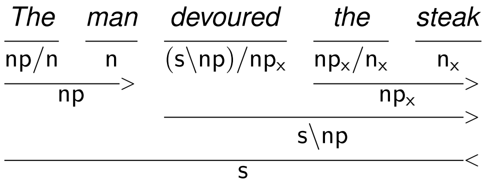


Figure 3: A simple CCG derivation.

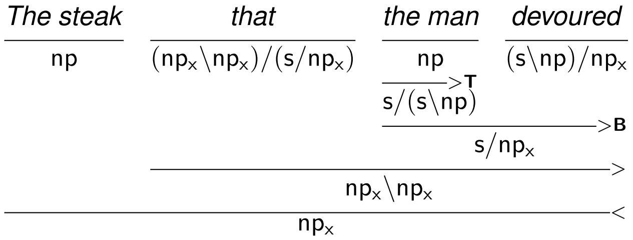


Figure 4: An example of CCG's treatment of relative clauses. The syntactic dependency between *devoured* and *steak* is the same as it was in figure 3. Co-indexations (the '*x*s') have been added here and above to aid the eye in following the relevant [*devoured-steak*] dependency.

(s\np)/np, or "the category that would become a sentence if it could combine with a noun phrase to the right and another noun phrase to the left". An example of how categories combine to make sentences is shown in Figure 3. CCG has many capabilities that go beyond that of a typical context-free grammar. First, it has a sophisticated internal system of managing syntactic heads and dependencies[1]. These dependencies are used to great effect in CCG-based semantic role labeling systems (Gildea and Hockenmaier, 2003; Boxwell et al., 2009), as they do not suffer the same data-sparsity effects encountered with treepath features in CFG-based SRL systems. Secondly, CCG permits these dependencies to be passed through intermediary categories in grammatical structures like relative clauses. In Figure 4, *the steak* is still in the object relation to *devoured*, even though the verb is inside a relative clause. Finally and most importantly, *these dependencies are represented directly on the CCG categories themselves*. This is what makes CCG resistant to the problem described in Section 3 – because the dependency is formed when the two heads combine, it is available to be used as a local feature by the semantic role labeler.

5 Semantic Role Labeling

We use a modified version of the Brutus semantic role labeling system (Boxwell et al., 2009)[2]. The original version of this system takes complete CCG derivations as input, and predicts semantic roles over them. For our purposes, however, it is necessary to modify the system to make semantic predictions at parse time, inside a packed chart, before the complete derivation is available. For this reason, it is necessary to remove the global features from the system (that is, features that rely on the complete parse), leaving only local features (features that are known at the moment that the predicate is attached to the argument). Crucially, dependency features count as "local" features, even though they have the potential to connect words that are very far apart in the sentence.

Brutus is arranged in a two-stage pipeline. First, a maximum entropy classifier[3] predicts, for each predicate in turn, which words in the sentence are likely headwords of semantic roles. Then, a second maximum entropy classifier assigns role labels to each of these words. The features used in the identification model of the local-only version of Brutus are as follows:

- **Words.** A three-word window surrounding the candidate word. For example, if we were considering the word *steak* in Figure 3, the three features would be represented as word_-1=the, word_0=steak, and word_1=#, with the last feature representing an out-of-bounds index.

- **Predicate.** The predicate whose semantic roles the system is looking for. For example, the sentence in figure 3 contains one predicate: *devour*.

- **Syntactic Dependency.** As with a previous approach in CCG semantic role labeling (Gildea and Hockenmaier, 2003), this feature shows the exact nature of the syntactic dependency between the predicate and the word we are considering, if any such dependency exists. This feature is represented by the category of the predicate, the argument slot that this word fits into, and whether or not the predicate is the head of the resultant category, represented with a left or right arrow. In the example from figure 3, the relationship between *devoured* and *steak* would be represented as (s\np)/np.2.→.

The second maximum entropy classifier uses all of the features from the identifier, plus several more:

[1] A complete explanation of CCG predicate-argument dependencies can be found in the CCGbank user manual (Hockenmaier and Steedman, 2005)

[2] Found at http://www.ling.ohio-state.edu/~boxwell/software/brutus.html

[3] Brutus uses Zhang Le's maxent toolkit, available at http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html.

Model	P	R	F
Local	89.8%	80.8%	85.1%
Global	89.8%	84.3%	87.0%

Table 1: SRL results for treebank parses, using the local model described in Section 5 and the full global model.

- **Before / After.** A binary indicator feature indicating whether the candidate word is before or after the predicate.

- **Result Category Detail.** This indicates the feature on the result category of the predicate. Possible values include dcl (for declarative sentences), pss (for passive sentences), ng (for present-progressive phrases like "running the race"), etc. These are read trivially off of the verbal category.

- **Argument Mapping.** An argument mapping is a prediction of a likely set of semantic roles for a given CCG predicate category. For example, a likely argument mapping for devoured:(s[dcl]\np)/np is [Arg0,Arg1]. These are predicted from string-level features, and are useful for bringing together otherwise independent classification decisions for individual roles. Boxwell et al. (2009) describe this feature in detail.

The Maximum-Entropy models were trained to 500 iterations. To prevent overfitting, we used Gaussian priors with global variances of 1 and 5 for the identifier and the labeler, respectively. Table 1 shows SRL performance for the local model described above, and the full global CCG-system described by Boxwell et al. (2009). We use the method for calculating the accuracy of Propbank verbal semantic roles described in the CoNLL-2008 shared task on semantic role labeling (Surdeanu et al., 2008). Because the Brutus SRL system is not designed to accommodate Nombank roles (Meyers et al., 2004), we restrict ourselves to predicting Propbank roles in the present work.

The local system has the same precision as the global one, but trails it on recall and F-measure. Note that this performance is achieved with gold standard parses.

6 Performing Semantic Role Predictions at Parse Time

Recall that the reasoning for using a substantially pared down version of the Brutus SRL system is to allow it to predict semantic roles in the context of a packed chart. Because we predict semantic roles for each constituent immediately after the constituent is formed and before it is added to the chart, we can use semantic roles to inform parsing. We use a CKY parsing algorithm, though this approach could be easily adapted to other parse strategies.

Whenever two constituents are combined, the SRL system checks to see if either of the constituents contains predicates. The system then attempts to identify semantic roles in the other constituent related to this predicate. This process repeats at every step, creating a combined syntax-semantics parse forest. Crucially, this allows us to use features derived from the semantic roles to rank parses inside the packed chart. This could result in an improvement over ranking completed parses, because reranking completed parses requires first generating an n-best list of parse candidates, potentially preventing the re-ranker from examining high value parses falling outside the n-best list.

In order to train our parse model, it is necessary to first employ a baseline parse model over the training set. The baseline model is a PCFG model, where the products of the probabilities of individual rule applications are used to rank candidate parses. We use a cross-fold validation technique to parse the training set (train on sections 02-20 to parse section 21, train on sections 02-19 and 21 to parse section 20, and so on). As we parse these sentences, we use the local SRL model described in Section 5 to predict semantic roles inside the packed chart. We then iterate over the packed chart and extract features based on the semantic roles in it, effectively learning from every possible semantic role in the parse forest. Notice that this does not require enumerating every parse in the forest (which would be prohibitively expensive) – the roles are labeled at parse time and can therefore be read directly from the packed chart. For each role in the packed chart, we label it as a "good" semantic role if it appears in the human-judged Propbank annotation for that sentence, and a "bad" semantic role if it does not.

The features extracted from the packed chart are as follows:

- **Role**. The semantic role itself, concatenated with the predicate. For example, play.Arg1. This will represent the intuition described in Section 1 that certain roles are more semantically appealing than others.

- **Role and Headword**. The semantic role concatenated with the predicate and the headword of the semantic role. This reflects the idea that certain words fit with particular roles better than others.

These features are used to train an averaged perceptron model to distinguish between likely and unlikely semantic roles. We incorporate the perceptron directly with the parser using a packed feature forest implementation, following an approach used by the current state-of-the-art CCG parser (Clark and Curran, 2004). By preferring sentences with good semantic roles, we hope to produce parses that give better overall semantic role predictions. The parser prefers spans with better semantic roles, and breaks ties that would have arisen using the baseline model alone. Similarly the baseline model can break ties between equivalent semantic roles; this has the added benefit of encouraging normal-form derivations in cases of spurious ambiguity. The result is a single-best complete parse with semantic roles already predicted. Once the single-best parse is selected, we allow the global SRL model to predict any additional roles over the parse, to catch those roles that are difficult to predict from local features alone.

7 Experiment 1: Choosing a Single-Best Derivation from an N-best List

Our first experiment demonstrates our model's performance in a ranking task. In this task, a list of candidate parses are generated by our baseline model. This baseline model treats rule applications as a PCFG – each rule application (say, np + s\np = s) is given a probability in the standard way. The rule probabilities are unsmoothed maximum likelihood estimates derived from rule counts in the training portion of CCGbank. After n-best derivations are produced by the baseline model, we use the Brutus semantic role labeler to assign roles to each candidate derivation. We vary the size of the n-best list from 1 to 10 (note that an n-best list of size 1 is equivalent to the single-best baseline parse). We then use the semantic model to re-rank the candidate parses and produce a single-best parse. The outcomes are shown in Table 2.

n	P	R	F
1	85.1	71.7	77.8
2	85.9	74.8	79.9
5	**84.5**	**76.8**	**80.5**
10	83.7	76.8	80.1
C&C	83.6	76.8	80.0

Table 2: SRL performance on the development set (section 00) for various values of n. The final row indicates SRL performance on section 00 parses from the Clark and Curran CCG parser.

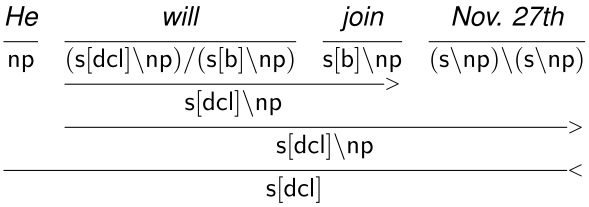


Figure 5: The single-best analysis for *He will join Nov 27th* according to the baseline model. Notice that the temporal adjunct is attached high, leading the semantic role labeler to fail to identify ArgM-TMP.

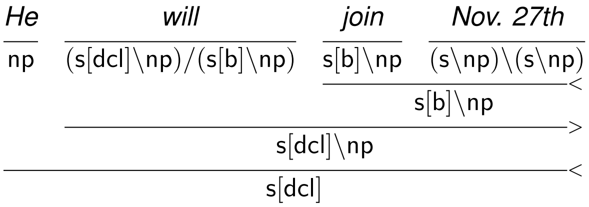


Figure 6: The second-best analysis of *He will join Nov 27th*. This analysis correctly predicts *Nov 27th* as the ArgM-TMP of *join*, and the semantic model correctly re-ranks this analysis to the single-best position.

The availability of even two candidate parses yields a 2.1% boost to the balanced F-measure. This is because the semantic role labeler is very sensitive to syntactic attachment decisions, and in many cases the set of rule applications used in the derivation are very similar or even the same. Consider the simplified version of a phenomenon found in wsj_0001.1 shown in Figures 5 and 6. The only difference in rule applications in these derivations is whether the temporal adjunct attaches to s[b]\np or s[dcl]\np. Because the s[dcl]\np case is slightly more common in the treebank, the baseline model identifies it as the single-best parse, and identifies the derivation in figure 6 as the second-best parse. The semantic model, however, correctly recognizes that the semantic roles predicted by the derivation in Figure 6 are superior to those predicted by the derivation in figure 5. This demonstrates how a second or third-best parse according to the baseline model can be greatly superior to the single-best in terms of semantics.

8 Experiment 2: Choosing a Single-Best Derivation Directly from the Packed Chart

One potential weakness with the n-best list approach described in Section 7 is choosing the size of the n-best list. As the length of the sentence grows, the number of candidate analyses grows. Because sentences in the treebank and in real-world applications are of varying length and complexity, restricting ourselves to an n-best list of a particular size opens us to considering some badly mangled derivations on short, simple sentences, and not enough derivations on long, complicated ones. One possible solution to this is to simply choose a single best derivation directly from the packed chart using the semantic model, eschewing the baseline model entirely except for breaking ties. In this approach, we use the local SRL model described in section 6 to predict semantic roles at parse time, inside the packed chart. This frees us from the need to have a complete derivation (as in the n-best list approach in Section 7). We use the semantic model to choose a single-best parse from the packed chart, then we pass this complete parse through the global SRL model to give it all the benefits afforded to the parses in the n-best approach. The results for the semantic model compared to the baseline model are shown in table 3. Interestingly, the semantic model performs considerably worse than the baseline model. To understand why, it is necessary to remember that the semantic model uses only semantic features – probabilities of rule applications are not considered. Therefore, the semantic model is perfectly happy to predict derivations with sequences of highly unlikely rule applications so long as they predict a role that the model has been trained to prefer.

Model	P	R	F
Baseline	85.1	71.7	77.8
Semantic	82.7	70.5	76.1

Table 3: A comparison of the performance of the baseline model and the semantic model on semantic role labeling. The semantic model, when unrestrained by the baseline model, performs substantially worse.

Apparently, the reckless pursuit of appealing semantic roles can ultimately harm semantic role labeling accuracy as well as parse accuracy. Consider the analysis shown in Figure 7. Because the averaged perceptron semantic model is not sensitive to the relationships between different semantic roles, and because Arg1 of *name* is a "good" semantic role, the semantic model predicts as many of them as it can. The very common np-appositive construction is particularly vulnerable to this kind of error, as it can be easily mistaken for a three-way coordination (like *carrots, peas and watermelon*). Many of the precision errors generated by the local model are of this nature, and the global model is unlikely to remove them, given the presence of strong dependencies between each of the "subjects" and the predicate.

Coordination errors are also common when dealing with relative clause attachment. Consider the analysis in Figure 8. To a PCFG model, there is little difference between attaching the relative clause to *the researchers* or *Lorillard nor the researchers*. The semantic model, however, would rather predict two semantic roles than just one (because study:Arg0 is a highly appealing semantic role). Once again, the pursuit of appealing semantic roles has led the system astray.

We have shown in Section 7 that the semantic model can improve SRL performance when it is constrained to the most likely PCFG derivations, but enumerating n-best lists is costly and cumbersome. We can, however, combine the semantic model with the baseline PCFG. Our method for doing this is designed to avoid the kinds of error described above. We first identify the highest-scoring parse according to the PCFG model. This parse will be used in later processing unless we are able to identify another parse that satisfies the following criteria:

1. It must be closely related to the parse that has the best score according to the semantic model. To identify such parses, we ask the chart unpacking algorithm to generate all the parses that can be reached by making up to five attachment changes to this semantically preferred parse – no more.

2. It must have a PCFG score that is not much less than that of the single-best PCFG parse. We do this by requiring that it has a score that is within a factor of α of the best available. That is, the single-best parse from the semantic model must satisfy

$$\log P(\text{sem}) > \log P(\text{baseline}) + \log(\alpha)$$

where the α value is tuned on the development set.

If no semantically preferred parse meets the above criteria, the single-best PCFG parse is used. We find that the PCFG-preferred parse is used about 35% of the time and an alternative used instead about 65% of the time. The SRL performance for this regime, using a range of cut-off factors, is shown in table 4. On this basis we select a cut-off of 0.5 as suitable for use for final testing. On the development set this method gives the best precision in extracting dependencies, but is slightly inferior to the method using a 2-best list on recall and balanced F-measure.

Factor (α)	P	R	F
0.5	86.3	71.9	78.5
0.1	85.4	72.0	78.1
0.05	85.2	72.0	78.0
0.005	84.3	71.3	77.3

Table 4: SRL accuracy when the semantic model is constrained by the baseline model

9 Results and Discussion

We use the method for calculating SRL performance described in the CoNNL 2008 and 2009 shared tasks. However, because the semantic role labeler we use was not designed to work with Nombank (and it is difficult to separate Nombank and Propbank predicates from the publicly released shared task output), it is not feasible to compare results with the candidate systems described there. We can, however, compare our two experimental models with our baseline parser and the current state-of-the-art CCG

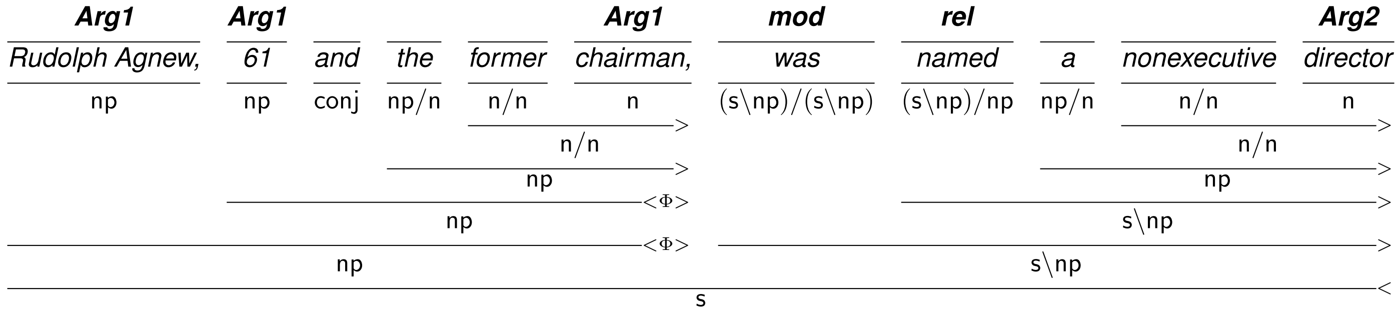


Figure 7: A parse produced by the unrestricted semantic model. Notice that *Rudolph Agnew, 61 and the former chairman* is erroneously treated as a three-way conjunction, assigning semantic roles to all three heads.

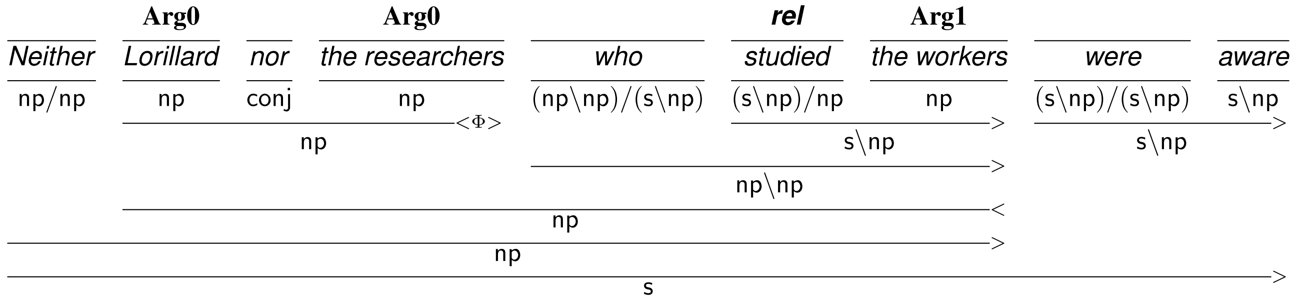


Figure 8: Relative clause attachment poses problems when preceded by a conjunction – the system generally prefers attaching relative clauses high. In this case, the relative clause should be attached low.

parser (Clark and Curran, 2004). The results on the test set (WSJ Section 23, <40 words) are shown in Table 5.

There are many areas for potential improvement for the system. The test set scores of both of our experimental models are lower than their development set scores,where the n-best model outperforms even the Clark and Curran parser in the SRL task. This may be due to vocabulary issues (we are of course unable to evaluate if the vocabulary of the training set more closely resembles the development set or the test set). If there are vocabulary issues, they could be alleviated by experimenting with POS based lexical features, or perhaps even generalizing a latent semantics over heads of semantic roles (essentially identifying broad categories of words that appear with particular semantic roles, rather than counting on having encountered that particular word in training). Alternately, this drop in performance could be caused by a mismatch in the average length of sentences, which would cause our α factor and the size of our n-best lists (which were tuned on the development set) to be suboptimal. We anticipate the opportunity to further explore better ways of determining n-best list size. We also anticipate the possibility of integrating the semantic model with a state-of-the-art CCG parser, potentially freeing the ranker from the limitations of a simple PCFG baseline.

It is also worth noting that the chart-based model seems heavily skewed towards precision. Because the parser can dig deeply into the chart, it is capable of choosing a parse that predicts only semantic roles that it is highly confident about. By choosing these parses (and not parses with less attractive semantic roles), the model can maximize the average score of the semantic roles it predicts. This tendency towards identifying only the most certain roles is consistent with high-precision low-recall results. The n-best parser has a much more restricted set of semantic roles from parses more closely resembling the single-best parse, and therefore is less likely to be presented with the opportunity to choose parses that do away with less likely (but still reasonable) roles.

10 Conclusions and Future Work

In this paper, we discuss the procedure for identifying semantic roles at parse time, and using these roles to guide the parse. We demonstrate that using semantic roles to guide parsing can improve overall SRL performance, but that these same benefits can be realized by re-ranking an n-best list with the same model. Regardless, there are several reasons why it is useful to have the ability to predict semantic roles inside the chart.

Predicting semantic roles inside the chart could be used to perform SRL on very long or unstructured passages.

	SRL			Labeled Deps		
Model	P	R	F	P	R	F
Baseline	84.7	70.7	77.0	80.0	79.8	79.9
Rank n=5	82.0	73.7	77.7	80.1	80.0	80.0
Chart	90.0	68.4	77.7	82.3	80.2	81.2
C&C	83.3	77.6	80.4	84.9	84.6	84.7
Char	77.1	75.5	76.5	-	-	-

Table 5: The full system results on the test set of the WSJ corpus (Section 23). Included are the baseline parser, the n-best reranking model from Section 7, the single-best chart-unpacking model from Section 8, and the state-of-the-art C&C parser. The final row shows the SRL performance obtained by Punyakanok et al. (2008) using the Charniak parser. Unfortunately, their results are evaluated based on spans of words (rather than headword labels), which interferes with direct comparison. The Charniak parser is a CFG-style parser, making labeled dependency non-applicable.

Most parsing research on the Penn Treebank (the present work included) focuses on sentences of 40 words or less, because parsing longer sentences requires an unacceptably large amount of computing resources. In practice, however, semantic roles are rarely very distant from their predicates – generally they are only a few words away; often they are adjacent. In long sentences, the prediction of an entire parse may be unnecessary for the purposes of SRL.

The CKY parsing algorithm works by first predicting all constituents spanning two words, then all constituents spanning three words, then four, and so on until it predicts constituents covering the whole sentence. By setting a maximum constituent size (say, ten or fifteen), we could abandon the goal of completing a spanning analysis in favor of identifying semantic roles in the neighborhood of their predicates, eliminating the need to unpack the chart at all. This could be used to efficiently perform SRL on poorly structured text or even spoken language transcriptions that are not organized into discrete sentences. Doing so would also eliminate the potentially noisy step of automatically separating out individual sentences in a larger text. Alternately, roles predicted in the chart could even be incorporated into a low-precision-high-recall information retrieval system seeking a particular semantic relationship by scanning the chart for a particular semantic role.

Another use for the packed forest of semantic roles could be to predict complete sets of roles for a given sentence using a constraint based method like integer linear programming. Integer linear programming takes a large number of candidate results (like semantic roles), and applies a set of constraints over them (like "roles may not overlap" or "no more than one of each role is allowed in each sentence") to find the optimal set. Doing so could eliminate the need to unpack the chart at all, effectively producing semantic roles without committing to a single syntactic analysis.

11 Acknowledgements

We would like to thank Mike White, William Schuler, Eric Fosler-Lussier, and Matthew Honnibal for their helpful feedback.

References

Stephen A. Boxwell, Dennis N. Mehay, and Chris Brew. 2009. Brutus: A semantic role labeling system incorporating CCG, CFG, and Dependency features. In *Proc. ACL-09*.

E. Charniak. 2001. Immediate-head parsing for language models. In *Proc. ACL-01*, volume 39, pages 116–123.

Stephen Clark and James R. Curran. 2004. Parsing the WSJ using CCG and Log-Linear Models. In *Proc. ACL-04*.

J.R. Finkel and C.D. Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 326–334. Association for Computational Linguistics.

Daniel Gildea and Julia Hockenmaier. 2003. Identifying semantic roles using Combinatory Categorial Grammar. In *Proc. EMNLP-03*.

J. Hajič, M. Ciaramita, R. Johansson, D. Kawahara, M.A. Martí, L. Màrquez, A. Meyers, J. Nivre, S. Padó, J. Štěpánek, et al. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–18. Association for Computational Linguistics.

J. Hockenmaier and M. Steedman. 2005. CCGbank manual. Technical report, MS-CIS-05-09, University of Pennsylvania.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

A. Meyers, R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman. 2004. The nombank project: An interim report. In A. Meyers, editor, *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, pages 24–31, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106.

Vasin Punyakanok, Dan Roth, and Wen tau Yih. 2008. The Importance of Syntactic Parsing and Inference in Semantic Role Labeling. *Computational Linguistics*, 34(2):257–287.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press.

M. Surdeanu, R. Johansson, A. Meyers, L. Màrquez, and J. Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings* *of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177. Association for Computational Linguistics.

Word Sense Induction & Disambiguation Using Hierarchical Random Graphs

Ioannis P. Klapaftis
Department of Computer Science
University of York
United Kingdom
giannis@cs.york.ac.uk

Suresh Manandhar
Department of Computer Science
University of York
United Kingdom
suresh@cs.york.ac.uk

Abstract

Graph-based methods have gained attention in many areas of Natural Language Processing (NLP) including Word Sense Disambiguation (WSD), text summarization, keyword extraction and others. Most of the work in these areas formulate their problem in a graph-based setting and apply unsupervised graph clustering to obtain a set of clusters. Recent studies suggest that graphs often exhibit a hierarchical structure that goes beyond simple flat clustering. This paper presents an unsupervised method for inferring the hierarchical grouping of the senses of a polysemous word. The inferred hierarchical structures are applied to the problem of word sense disambiguation, where we show that our method performs significantly better than traditional graph-based methods and agglomerative clustering yielding improvements over state-of-the-art WSD systems based on sense induction.

1 Introduction

A number of NLP problems can be cast into a graph-based framework, in which entities are represented as vertices in a graph and relations between them are depicted by weighted or unweighted edges. For instance, in unsupervised WSD a number of methods (Widdows and Dorow, 2002; Véronis, 2004; Agirre et al., 2006) have constructed word co-occurrence graphs for a target polysemous word and applied graph-clustering to obtain the clusters (senses) of that word.

Similarly in text summarization, Mihalcea (2004) developed a method, in which sentences are represented as vertices in a graph and edges between them are drawn according to their common tokens or words of a given POS category, e.g. nouns. Graph-based ranking algorithms, such as PageRank (Brin and Page, 1998), were then applied in order to determine the significance of sentences. In the same vein, graph-based methods have been applied to other problems such as determining semantic similarity of text (Ramage et al., 2009).

Recent studies (Clauset et al., 2006; Clauset et al., 2008) suggest that graphs exhibit a hierarchical structure (e.g. a binary tree), in which vertices are divided into groups that are further subdivided into groups of groups, and so on, until we reach the leaves. This hierarchical structure provides additional information as opposed to flat clustering by explicitly including organisation at all scales of a graph (Clauset et al., 2008). In this paper, we present an unsupervised method for inferring the hierarchical structure (binary tree) of a graph, in which vertices are the contexts of a polysemous word and edges represent the similarity between contexts. The method that we use to infer that hierarchical structure is the Hierarchical Random Graphs (HRGs) algorithm due to Clauset et al. (2008).

The binary tree produced by our method groups the contexts of a polysemous word at different heights of the tree. Thus, it induces the senses of that word at different levels of sense granularity. To evaluate our method, we apply it to the problem of noun sense disambiguation showing that inferring the hierarchical structure using HRGs provides additional information from the observed graph leading to improved WSD performance compared to: (1)

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 745–755,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

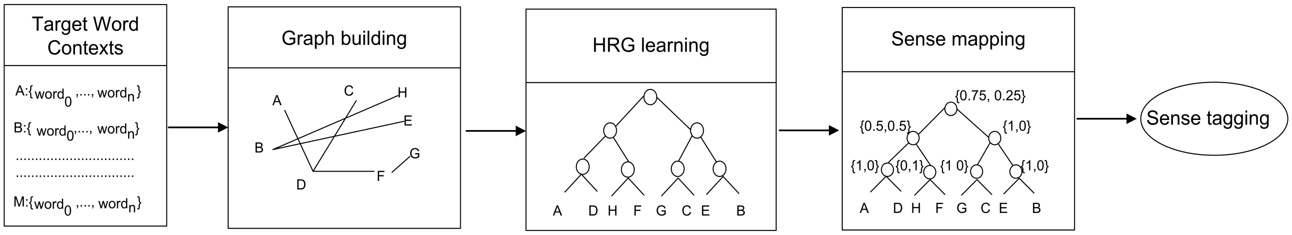


Figure 1: Stages of the proposed method.

simple flat clustering, and (2) traditional agglomerative clustering. Finally, we compare our results with state-of-the-art sense induction systems and show that our method yields improvements. Figure 1 shows the different stages of the proposed method that we describe in the following sections.

2 Related work

Typically, graph-based methods, when applied to unsupervised sense disambiguation represent each word w_i co-occurring with the target word tw as a vertex. Two vertices are connected via an edge if they co-occur in one or more contexts of tw. Once the co-occurrence graph of tw has been constructed, different graph clustering algorithms are applied to induce the senses. Each cluster (induced sense) consists of a set of words that are semantically related to the particular sense. Figure 2 shows an example of a graph for the target word *paper* that appears with two different senses *scholarly article* and *newspaper*.

Véronis (2004) has shown that co-occurrence graphs are small-world networks that contain highly dense subgraphs representing the different clusters (senses) of the target word (Véronis, 2004). To identify these dense regions Véronis's algorithm iteratively finds their hubs, where a hub is a vertex with a very high *degree*. The degree of a vertex is defined to be the number of edges incident to that vertex. The identified hub is then deleted along with its direct neighbours from the graph producing a new cluster.

For example, in Figure 2 the highest degree vertex, *news*, is the first hub, which would be deleted along with its direct neighbours. The deleted region corresponds to the *newspaper* sense of the target word *paper*. Véronis (2004) further processed the identified clusters (senses), in order to assign the rest of graph vertices to the identified clusters by utilising the *minimum spanning tree* of the original graph.

In Agirre et al. (2006), the algorithm of Véronis (2004) is analysed and assessed on the SensEval-3 dataset (Snyder and Palmer, 2004), after optimising its parameters on the SensEval-2 dataset (Edmonds and Dorow, 2001). The results show that the WSD F-Score outperforms the Most Frequent Sense (MFS) baseline by approximately 10%, while inducing a large number of clusters (with averages of 60 to 70).

Another graph-based method is presented in (Dorow and Widdows, 2003). They extract only noun neighbours that appear in conjunctions or disjunctions with the target word. Additionally, they extract second-order co-occurrences. Nouns are represented as vertices, while edges between vertices are drawn, if their associated nouns co-occur in conjunctions or disjunctions more than a given number of times. This co-occurrence frequency is also used to weight the edges. The resulting graph is then pruned by removing the target word and vertices with a low degree. Finally, the MCL algorithm (Dongen, 2000) is used to cluster the graph and produce a set of clusters (senses) each one consisting of a set of contextually related words.

Chinese Whispers (CW) (Biemann, 2006) is a parameter-free[1] graph clustering method that has been applied in sense induction to cluster the co-occurrence graph of a target word (Biemann, 2006), as well as a graph of collocations related to the target word (Klapaftis and Manandhar, 2008). The evaluation of the collocational-graph method in the SemEval-2007 sense induction task (Agirre and Soroa, 2007) showed promising results.

All the described methods for sense induction ap-

[1] One needs to specify only the number of iterations. The number of clusters is generated automatically.

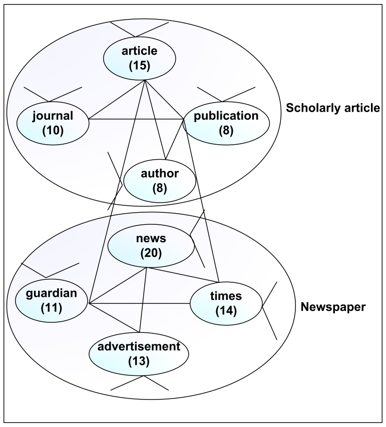
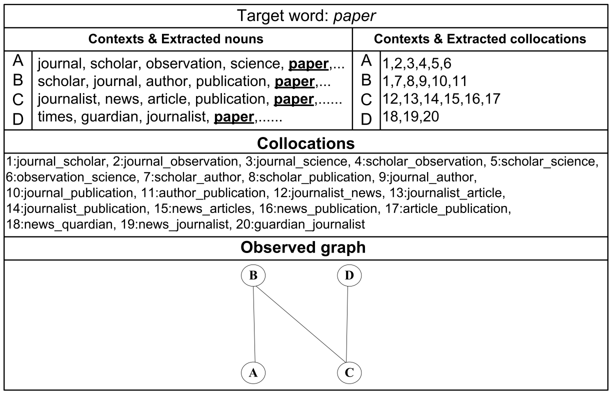


Figure 2: Graph of words for the target word *paper*. Numbers inside vertices correspond to their degree.



Target word: *paper*

	Contexts & Extracted nouns		Contexts & Extracted collocations
A	journal, scholar, observation, science, **paper**,...	A	1,2,3,4,5,6
B	scholar, journal, author, publication, **paper**,...	B	1,7,8,9,10,11
C	journalist, news, article, publication, **paper**,......	C	12,13,14,15,16,17
D	times, guardian, journalist, **paper**,......	D	18,19,20

Collocations

1:journal_scholar, 2:journal_observation, 3:journal_science, 4:scholar_observation, 5:scholar_science, 6:observation_science, 7:scholar_author, 8:scholar_publication, 9:journal_author, 10:journal_publication, 11:author_publication, 12:journalist_news, 13:journalist_article, 14:journalist_publication, 15:news_articles, 16:news_publication, 17:article_publication, 18:news_quardian, 19:news_journalist, 20:guardian_journalist

Observed graph

Figure 3: Running example of graph creation

ply flat graph clustering methods to derive the clusters (senses) of a target word. As a result, they neglect the fact that their constructed graphs often exhibit a hierarchical structure that is useful in several tasks including word sense disambiguation.

3 Building a graph of contexts

This section describes the process of creating a graph of contexts for a polysemous target word. Figure 3 provides a running example of the different stages of our method. In the example, the target word *paper* appears with the *scholarly article* sense in the contexts *A*, *B*, and with the *newspaper* sense in the contexts *C* and *D*.

3.1 Corpus preprocessing

Let bc denote the base corpus consisting of the contexts containing the target word tw. In our work, a context is defined as a paragraph[2] containing the target word.

The aim of this stage is to capture nouns contextually related to tw. Initially, the target word is removed from bc and part-of-speech tagging is applied to each context. Following the work in (Véronis, 2004; Agirre et al., 2006) only nouns are kept and lemmatised. In the next step, the distribution of each noun in the base corpus is compared to the distribution of the same noun in a reference corpus[3] using the log-likelihood ratio (G^2) (Dunning, 1993). Nouns with a G^2 below a pre-specified threshold (parameter p_1) are removed from each paragraph of the base corpus. The upper left part of Figure 3 shows the words kept as a result of this stage.

3.2 Graph creation

Graph vertices: To create the graph of vertices, we represent each context c_i as a vertex in a graph G.

Graph edges: Edges between the vertices of the graph are drawn based on their similarity, defined in Equation 1, where $sim_{cl}(c_i, c_j)$ is the *collocational weight* of contexts c_i, c_j and $sim_{wd}(c_i, c_j)$ is their bag-of-words weight. If the edge weight $W(c_i, c_j)$ is above a prespecified threshold (parameter p_3), then an edge is drawn between the corresponding vertices in the graph.

$$W(c_i, c_j) = \frac{1}{2}(sim_{cl}(c_i, c_j) + sim_{wd}(c_i, c_j)) \quad (1)$$

Collocational weight: The limited polysemy of collocations can be exploited to compute the similarity between contexts c_i and c_j. In our setting, a collocation is a juxtaposition of two nouns within the same context. Thus, given a context c_i, each of its nouns is combined with any other noun yielding a total of $\binom{N}{2}$ collocations for a context with N nouns. Each collocation, cl_{ij} is weighted using the log-likelihood ratio (G^2) (Dunning, 1993) and is filtered out if the G^2 is below a prespecified threshold (parameter p_2). At the end of this process, each context c_i of tw is associated with a vector of collocations (v_i). The upper right part of Figure 3 shows the collocations associated with each context of our example.

[2] Our definition of *context* is equivalent to an instance of the target word in the SemEval-2007 sense induction task dataset (Agirre and Soroa, 2007).

[3] The British National Corpus, 2001, Distributed by Oxford University Computing Services.

Given two contexts c_i and c_j, we calculate their collocational weight using the Jaccard coefficient on the collocational vectors, i.e. $sim_{cl}(c_i, c_j) = \frac{|v_i \cap v_j|}{|v_i \cup v_j|}$. The selection of Jaccard is based on the work of Weeds et al. (2004), who analyzed the variation in a word's distributionally nearest neighbours with respect to a variety of similarity measures. Their analysis showed that there are three classes of measures, i.e. those selecting distributionally more general neighbours (e.g. cosine), those selecting distributionally less general neighbours (e.g. AMCRM-Precision (Weeds et al., 2004)) and those without a bias towards the distributional generality of a neighbour (e.g. Jaccard). In our setting, we are interested in calculating the similarity between two contexts without any bias. We selected Jaccard, since the rest of that class's measures are based on pointwise mutual information that assigns high weights to infrequent events.

Bag-of-words weight: Estimating context similarity using collocations may provide reliable estimates regarding the existence of an edge in the graph, however, it also suffers from data sparsity. For this reason, we also employ a bag-of-words model. Specifically, each context c_i is associated with a vector g_i that contains the nouns kept as result of the corpus preprocessing stage. The upper left part of Figure 3 shows the words associated with each context of our example. Given two contexts c_i and c_j, we calculate their bag-of-words weight using the Jaccard coefficient on the word vectors, i.e. $sim_{wd}(c_i, c_j) = \frac{|g_i \cap g_j|}{|g_i \cup g_j|}$.

The collocational weight and bag-of-words weight are averaged to derive the edge weight between two contexts as defined in Equation 1. The resulting graph of our running example is shown on the bottom of Figure 3. This graph is the input to the *hierarchical random graphs* method (Clauset et al., 2008) described in the next section.

4 Hierarchical Random Graphs for sense induction

In this section, we describe the process of inferring the hierarchical structure of the graph of contexts using *hierarchical random graphs* (Clauset et al., 2008).

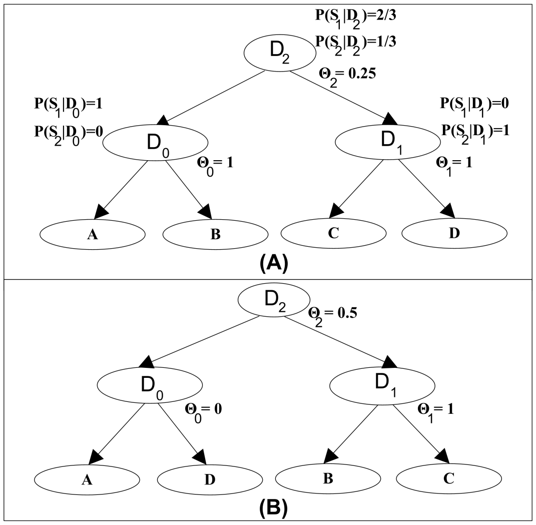


Figure 4: Two dendrograms for the graph in Figure 3.

4.1 The Hierarchical Random Graph model

A dendrogram is a binary tree with n leaves and $n-1$ parents. Figure 4 shows an example of two dendrograms with 4 leaves and 3 parents. Given a set of n contexts that we need to arrange hierarchically, let us denote by $G = (V, E)$ the graph of contexts, where $V = \{v_0, v_1 \dots v_n\}$ is the set of vertices, $E = \{e_0, e_1 \dots e_m\}$ is the set of edges and $e_k = \{v_i, v_j\}$.

Given an undirected graph G, each of its n vertices is a leaf in a dendrogram, while the internal nodes of that dendrogram indicate the hierarchical relationships among the leaves. We denote this organisation by $D = \{D_1, D_2, \dots D_{n-1}\}$, where each D_k is an internal node. Every pair of nodes (v_i, v_j) is associated with a unique D_k, which is their lowest common ancestor in the tree. In this manner D partitions the edges that exist in G.

The primary assumption in the hierarchical random graph model is that edges in G exist independently, but with a probability that is not identically distributed. In particular, the probability that an edge $\{v_i, v_j\}$ exists in G is given by a parameter θ_k associated with D_k, the lowest common ancestor of v_i and v_j in D. In this manner, the topological structure D and the vector of probabilities $\vec{\theta}$ define the HRG given by $H(D, \vec{\theta})$ (Clauset et al., 2008).

4.2 HRG parameterisation

Assuming a uniform prior over all HRGs, the target is to identify the parameters of D and $\vec{\theta}$, so that the chosen HRG is statistically similar to G. Let D_k be an internal node of dendrogram D and $f(D_k)$ be the number of edges between the vertices of the subtrees of the subtree rooted at D_k that actually exist in G. For example, in Figure 4(A), $f(D_2) = 1$, because there is one edge in G connecting vertices B and C. Let $l(D_k)$ be the number of leaves in the left subtree of D_k, and $r(D_k)$ be the number of leaves in the right subtree. For example in Figure 4(A), $l(D_2) = 2$ and $r(D_2) = 2$. The likelihood of the hierarchical random graph $(D, \vec{\theta})$ is defined in Equation 2, where $A(D_k) = l(D_k)r(D_k) - f(D_k)$.

$$L(D, \vec{\theta}) = \prod_{D_k \in D} \theta_k^{f(D_k)} (1 - \theta_k)^{A(D_k)} \qquad (2)$$

The probabilities θ_k that maximise the likelihood of a dendrogram D can be easily estimated using the method of MLE i.e $\overline{\theta}_k = \frac{f(D_k)}{l(D_k)r(D_k)}$. Substituting this into Equation 2 yields Equation 3. For numerical reasons, it is more convenient to work with the logarithm of the likelihood which is defined in Equation 4, where $h(\overline{\theta}_k) = -\overline{\theta}_k \log \overline{\theta}_k - (1 - \overline{\theta}_k) \log (1 - \overline{\theta}_k)$.

$$L(D) = \prod_{D_k \in D} [\overline{\theta}_k^{\overline{\theta}_k} (1 - \overline{\theta}_k)^{1 - \overline{\theta}_k}]^{l(D_k)r(D_k)} \qquad (3)$$

$$\log L(D) = - \sum_{D_k \in D} h(\overline{\theta}_k) l(D_k) r(D_k) \qquad (4)$$

As can be observed, each term $-l(D_k)r(D_k)h(\overline{\theta}_k)$ is maximised when θ_k approaches 0 or 1. This means that high-likelihood dendrograms partition vertices into subtrees, such that the connections among their vertices in the observed graph are either very rare or very common (Clauset et al., 2008). For example, consider the two dendrograms in Figures 4(A) and 4(B). We observe that 4(A) is more likely than 4(B), since it provides a better division of the network leaves. Particularly, the likelihood of 4(A) is $L(D_1) = (1^1 \cdot (1-1)^1) \cdot (1^1 \cdot (1-1)^1) \cdot (0.25^1 \cdot (1-0.25)^3) = 0.105$, while the likelihood of 4(B) is $L(D_2) = (0^0 \cdot (1-0)^1) \cdot (1^1 \cdot (1-1)^1) \cdot (0.5^2 \cdot (1-0.5)^2) = 0.062$.

4.2.1 MCMC sampling

Finding the values of θ_k using the MLE method is straightforward. However, this is not the case for maximising the likelihood function over the space of all possible dendrograms. Given a graph with n vertices, i.e. n leaves in each dendrogram, the total number of different dendrograms is super-exponential ($(2n-3)!! \approx \sqrt{2}(2n)^{n-1}e^{-n}$) (Clauset et al., 2006).

To deal with this problem, we use a Markov Chain Monte Carlo (MCMC) method that samples dendrograms from the space of dendrogram models with probability proportional to their likelihood. Each time MCMC samples a dendrogram with a new highest likelihood, that dendrogram is stored. Hence, our goal is to choose the highest likelihood dendrogram once MCMC has converged.

Following the work in (Clauset et al., 2008), we pick a set of transitions between dendrograms, where a transition is a re-arrangement of the subtrees of a dendrogram. In particular, given a current dendrogram D_{curr}, each internal node D_k of D_{curr} is associated with three subtrees of D_{curr}. For instance, in Figure 5A, the subtrees st_1 and st_2 are derived from the two children of D_k and the third st_3 from its sibling. Given a current dendrogram, D_{curr}, the algorithm proceeds as follows:

1. Choose an internal node, $D_k \in D_{curr}$ uniformly.

2. Generate two possible new configurations of the subtrees of D_k (See Figure 5).

3. Choose one of the configurations uniformly to generate a new dendrogram, D_{next}.

4. Accept or reject D_{next} according to Metropolis-Hastings (MH) rule.

5. If transition is accepted, then $D_{curr} = D_{next}$.

6. GOTO 1.

According to MH rule (Newman and Barkema, 1999), a transition is accepted if $\log L(D_{next}) \geq \log L(D_{curr})$; otherwise the transition is accepted with probability $\frac{L(D_{next})}{L(D_{curr})}$. These transitions define an ergodic Markov chain, hence its stationary distribution can be reached (Clauset et al., 2008).

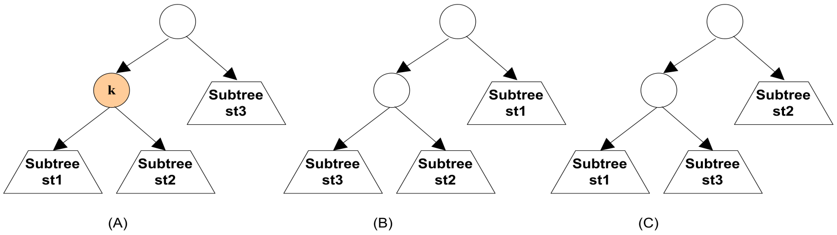


Figure 5: (A) current configuration for internal node D_k and its associated subtrees (B) first alternative configuration, (C) second alternative configuration. Note that swapping $st1$, $st2$ in (A) results in an equivalent tree. Hence, this configuration is excluded.

In our experiments, we noticed that the algorithm converged relatively quickly. The same behaviour (roughly $O(n^2)$ steps) was also noticed in Clauset et al. (2008), when considering graphs with thousands of vertices.

5 HRGs for sense disambiguation

5.1 Sense mapping

The output of HRG learning is a dendrogram D with n leaves (contexts) and $n-1$ internal nodes. To perform sense disambiguation, we mapped the internal nodes to gold standard senses using a sense-tagged corpus. Such a sense-tagged corpus is needed when induced word senses need to be mapped to a gold standard sense inventory.

Instead of using a hard mapping from the dendrogram internal nodes to the Gold Standard (GS) senses, we use a soft probabilistic mapping and calculate $P(s_k|D_i)$, i.e the probability of sense s_k given node D_i. Let $F(D_i)$ be the set of training contexts grouped by internal node D_i. Let $F'(s_k)$ be the set of training contexts that are tagged with sense s_k. Then the conditional probability, $P(s_k|D_i)$, is defined in Equation 5.

$$P(s_k|D_i) = \frac{|F(D_i) \cap F'(s_k)|}{|F(D_i)|} \quad (5)$$

Table 1 provides a sense-tagged corpus for the running example of Figure 3. Using this corpus and the tree in Figure 4(A), $P(s_1|D_2) = \frac{2}{3}$ and $P(s_2|D_2) = \frac{1}{3}$. In Figure 4(A) the rest of the calculated conditional probabilities are given.

5.2 Sense tagging

For evaluation we compared the proposed method against the current state-of-the-art sense induction systems in the WSD task. We followed the setting of SemEval-2007 sense induction task (Agirre and Soroa, 2007). In this setting, the base corpus (bc) (Section 3.1) for a target word consists both of the training and testing corpus. As a result, a testing context c_j of tw is a leaf in the generated dendrogram. The process of disambiguating c_j is straightforward exploiting the structural information provided by HRGs.

GS sense	Context ID	Context words
s_1	A	journal, scholar, observation science, **paper**
s_1	B	scholar, scholar, author, publication, **paper**
s_2	D	times, guardian, journalist, **paper**

Table 1: Sense-tagged corpus for the example in Figure 3

$$w(s_k, c_j) = \sum_{D_i \in H(c_j)} P(s_k|D_i) \cdot \theta_i \quad (6)$$

$$w(s^*, c_j) = \text{argmax}\, s_k(w(s_k, c_j)) \quad (7)$$

Let $H(c_j)$ denote the set of parents for context c_j. Then, the weight assigned to sense s_k is the sum of weighted scores provided by each identified parent. This is shown in Equation 6, where θ_i is the probability associated with each internal node D_i from the hierarchical random graph (see Figure 4(A)). This probability reflects the discriminating ability of internal nodes.

Finally, the highest weight determines the winning sense for context c_j (Equation 7). In our example (Figure 4(A)), $w(s_1, C) = (0 \cdot 1 + \frac{2}{3} \cdot 0.25) = 0.16$ and $w(s_2, C) = (1 \cdot 1 + \frac{1}{3} \cdot 0.25) = 1.08$. Hence, s_2 is the winning sense.

Parameter	Range
G^2 word threshold (p_1)	15,25,35,45
G^2 collocation threshold (p_2)	10,15,20
Edge similarity threshold (p_3)	0.05,0.09,0.13

Table 2: Parameter values used in the evaluation.

6 Evaluation

6.1 Evaluation setting & baselines

We evaluate our method on the nouns of the SemEval-2007 word sense induction task (Agirre and Soroa, 2007) under the second evaluation setting of that task, i.e. supervised evaluation. Specifically, we use the standard WSD measures of precision and recall in order to produce their harmonic mean (F-Score). The official scoring software of that task has been used in our evaluation. Note that the unsupervised measures of that task are not directly applicable to our induced hierarchies, since they focus on assessing flat clustering methods.

The first aim of our evaluation is to test whether inferring the hierarchical structure of the constructed graphs improves WSD performance. For that reason our first baseline, Chinese Whispers Unweighted version (*CWU*), takes as input the same unweighted graph of contexts as HRGs in order to produce a flat clustering. The set of produced clusters is then mapped to GS senses using the training dataset and performance is then measured on the testing dataset. We followed the same sense mapping method as in the SemEval-2007 sense induction task (Agirre and Soroa, 2007).

Our second baseline, Chinese Whispers Weighted version (*CWW*), is similar to the previous one, with the difference that the edges of the input graph are weighted using Equation 1. For clustering the graphs of *CWU* and *CWW* we employ, *Chinese Whispers*[4] (Biemann, 2006).

The second aim of our evaluation is to assess whether the hierarchical structure inferred by HRGs is more informative than the hierarchical structure inferred by traditional Hierarchical Clustering (*HAC*). Hence, our third baseline, takes as input a similarity matrix of the graph vertices and performs bottom-up clustering with average-linkage, which has already been used in WSI in (Pantel and Lin, 2003) and was shown to have superior or similar performance to single-linkage and complete-linkage in the related problem of learning a taxonomy of senses (Klapaftis and Manandhar, 2010).

To calculate the similarity matrix of vertices we follow a process similar to the one used in Section 4.2 for calculating the probability of an internal node. The similarity between two vertices is calculated according to the degree of connectedness among their direct neighbours. Specifically, we would like to assign high similarity to pairs of vertices, whose neighbours are close to forming a clique.

Given two vertices (contexts) c_i and c_j, let $N(c_i, c_j)$ be the set of their neighbours and $K(c_i, c_j)$ be the set of edges between the vertices in $N(c_i, c_j)$. The maximum number of edges that could exist between vertices in $N(c_i, c_j)$ is $\binom{|N(c_i,c_j)|}{2}$. Thus, the similarity of c_i, c_j is set equal to the number of edges that actually exist in that neighbourhood divided by the total number of edges that could exist ($\frac{|K(c_i,c_j)|}{\binom{|N(c_i,c_j)|}{2}}$).

The disambiguation process using the HAC tree is identical to the one presented in Section 5.2 with the only difference that the internal probability, θ_i, in Equation 6 does not exist for HAC. Hence, we replaced it with the factor $\frac{1}{|H(D_i)|}$, where $H(D_i)$ is the set of children of internal node D_i. This factor provides lower weights for nodes high in the tree, since their discriminating ability will possibly be lower.

6.2 Results & discussion

Table 2 shows the parameter values used in the evaluation. Figure 6(A) shows the performance of the proposed method against the baselines for $p_3 = 0.05$ and different p_1 and p_2 values. Figure 6(B) illustrates the results of the same experiment using $p_3 = 0.09$. In both figures, we observe that *HRGs* outperform the *CWU* baseline under all parameter combinations. In particular, all of the 12 performance differences for $p_3 = 0.09$ are statistically significant using McNemar's test at 95% confidence level, while for $p_3 = 0.05$ only 2 out of the 12 performance differences were not judged as significant from the test.

The picture is the same for $p_3 = 0.13$, where *CWU* performs significantly worse than for $p_3 =$

[4] The number of iterations for CW was set to 200.

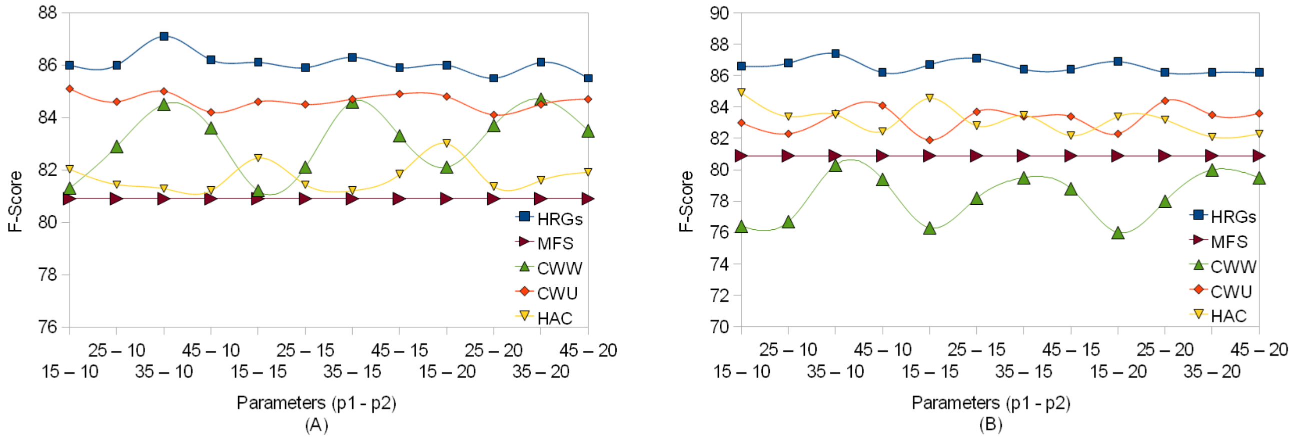


Figure 6: Performance analysis of HRGs, CWU, CWW & HAC for different parameter combinations (Table 2). **(A)** All combinations of p_1, p_2 and $p_3 = 0.05$. **(B)** All combinations of p_1, p_2 and $p_3 = 0.09$.

0.05 and $p_3 = 0.09$. Specifically, the largest performance difference between *HRGs* and *CWU* is 9.4% at $p_1 = 25$, $p_2 = 10$ and $p_3 = 0.13$. Setting the vertex similarity threshold (p_3) equal to 0.13 leads to more sparse and disconnected graphs, which causes *Chinese Whispers* to produce a large number of clusters. This leads to sparsity problems and unreliable mapping of clusters to GS senses due to the lack of adequate training data. In contrast, *HRGs* suffer less at this high threshold, although their performance when $p_3{<}0.13$ is better.

This picture does not change for the weighted version of *Chinese Whispers* (*CWW*) which performs worse than *CWU*. This is because *CWW* produces a smaller number of clusters than *CWU* that conflate the target word senses. It seems that using weighted edges creates a bias towards the MFS, in effect missing rare senses of a target word. This means that a number of words in the bag-of-words context vectors and collocations in the collocational context vectors (Section 3.2) are associated to more than one sense of the target word and most strongly associated to the MFS. As a result, increasing the p_1 threshold to 25 and 35 leads to a higher performance for *CWW*, since many of these words and collocations are filtered out.

Overall, the comparison of *HRGs* against the *CWU* and *CWW* baselines has shown that inferring the hierarchical structure of observed graphs leads to improved WSD performance as opposed to using flat clustering. This is because *HRGs* are able to infer both the hierarchical structure of the graph and include the probabilities, θ_k, associated with each internal node. These probabilities reflect the discriminating ability of each node, offering information missed by flat clustering.

In Figures 6(A) and 6(B) we observe that *HRGs* perform significantly better than HAC. In particular, all of their performance differences are statistically significant for these parameter values. The largest performance difference is 6.0% at $p_1 = 45$, $p_2 = 10$ and $p_3 = 0.05$. However, this picture is not the same when considering a higher context similarity threshold ($p_3 = 0.13$) as Figure 7 shows. In particular, *HRGs* and *HAC* perform similarly for $p_3 = 0.13$, while the majority of performance differences are not statistically significant.

The similar behaviour of *HRGs* and *HAC* at this threshold is caused both by the worse performance of *HRGs* and the improved performance of *HAC* as opposed to lower p_3 values. As it has been mentioned, setting $p_3 = 0.13$ leads to sparse and disconnected graphs. Additionally, the likelihood function (Equation 3) is maximised when the probability, θ_k, of an internal node, D_k, approaches 0 or 1. This creates a bias towards dendrograms, in which a large number of internal nodes have zero probability. These dendrograms might be a good-fit to the observed graph, but not to the GS.

In contrast, *HAC* is less affected, because it never considers creating an internal node, when the maximum similarity among any pair of two candidate

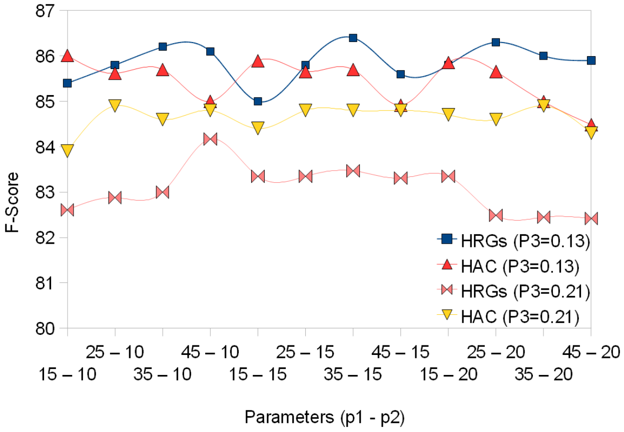


Figure 7: Performance of *HRGs* and *HAC* for different parameter combinations (Table 2). All combinations of p_1, p_2 and $p_3 \geq 0.13$.

subtrees is zero. Additionally, our experiments show that *HAC* is unable to deal with noise when considering sparse graphs (p_3<0.13). For that reason, the F-Score of *HAC* increases as the edge similarity threshold decreases.

To further investigate this issue and test whether *HAC* is able to achieve a higher F-Score than HRGs in higher p_3 values, we executed two more experiments for *HAC* and *HRGs* increasing p_3 to 0.17 and 0.21 respectively. In the first case we observed that the performance of *HAC* remained relatively stable compared to $p_3 = 0.13$, while in the second case the performance of *HAC* decreased as Figure 7 shows. In both cases, *HAC* performed significantly better than *HRGs*.

Overall, the comparison of *HRGs* against *HAC* has shown that *HRGs* perform significantly better than *HAC* when considering connected or less sparse graphs (p_3<0.13). This is due to the fact that *HAC* creates dendrograms, in which connections within the clusters are dense, while connections between the clusters are sparse, i.e. it only considers assortative structures. In contrast, HRGs also consider disassortative dendrograms, i.e. dendrograms in which vertices are less likely to be connected on small scales than on large ones, as well as mixtures of assortative and disassortative (Clauset et al., 2008). This is achieved by allowing the probability θ_k of a node k to vary arbitrarily throughout the dendrogram.

HAC performs similarly or better than *HRGs* for largely disconnected and sparse graphs, because HRGs become biased towards disassortative trees which are not a good fit to the GS (Figure 7). Despite that, our evaluation has also shown that the best performance of *HAC* (F-Score = 86.0% at $p_1 = 15$, $p_2 = 10$, $p_3 = 0.13$) is significantly lower than the best performance of *HRGs* (F-Score = 87.6% at $p_1 = 35$, $p_2 = 10$, $p_3 = 0.09$).

6.3 Comparison to state-of-the-art methods

Table 3 compares the best performing parameter combination of our method against state-of-the-art methods. Table 3 also includes the best performance of our baselines, i.e *HAC*, *CWU* and *CWW*.

Brody & Lapata (2009) presented a sense induction method that is related to Latent Dirichlet Allocation (Blei et al., 2003). In their work, they model the target word instances as samples from a multinomial distribution over senses which are successively characterized as distributions over words (Brody and Lapata, 2009). A significant advantage of their method is the inclusion of more than one layer in the LDA setting, where each layer corresponds to a different feature type e.g. dependency relations, bigrams, etc. The inclusion of different feature types as separate models in the sense induction process can easily be modeled in our setting, by inferring a different hierarchy of target word instances according to each feature type, and then combining all of them to a consensus tree. In this work, we have focused on extracting a single hierarchy combining word co-occurrence and bigram features.

Niu et al. (2007) developed a vector-based method that performs sense induction by grouping the contexts of a target word using three types of features, i.e. POS tags of neighbouring words, word co-occurrences and local collocations. The *sequential information bottleneck* algorithm (Slonim et al., 2002) is applied for clustering. *HRGs* perform slightly better than the methods of Brody & Lapata (2009) and Niu et al. (2007), although the differences are not significant (McNemar's test at 95% confidence level).

Klapaftis & Manandhar (2008) developed a graph-based sense induction method, in which vertices correspond to collocations related to the target word and edges between vertices are drawn ac-

System	Performance (%)
HRGs	87.6
(Brody and Lapata, 2009)	87.3
(Niu et al., 2007)	86.8
(Klapaftis and Manandhar, 2008)	86.4
HAC	86.0
CWU	85.1
CWW	84.7
(Pedersen, 2007)	84.5
MFS	80.9

Table 3: HRGs against recent methods & baselines.

cording to the co-occurrence frequency of the corresponding collocations. The constructed graph is smoothed to identify more edges between vertices and then clustered using Chinese Whispers (Biemann, 2006). This method is related to the basic inputs of our presented method. Despite that, it is a flat clustering method that ignores the hierarchical structure exhibited by observed graphs. The previous section has shown that inferring the hierarchical structure of graphs leads to superior WSD performance.

Pedersen (2007) presented *SenseClusters*, a vector-based method that clusters second order co-occurrence vectors using k-means, where k is automatically determined using the Adapted Gap Statistic (Pedersen and Kulkarni, 2006). As can be observed, *HRGs* perform significantly better than the methods of Pedersen (2007) and Klapaftis & Manandhar (2008) (McNemar's test at 95% confidence level).

Finally, Table 3 shows that the best performing parameter combination of HRGs achieves a significantly higher F-Score than the best performing parameter combination of *HAC*, *CWU* and *CWW*. Furthermore, HRGs outperform the most frequent sense baseline by 6.7%.

7 Conclusion & future work

We presented an unsupervised method for inferring the hierarchical grouping of the senses of a polysemous word. Our method creates a graph, in which vertices correspond to contexts of a polysemous target word and edges between them are drawn according to their similarity. The *hierarchical random graphs* algorithm (Clauset et al., 2008) was applied to the constructed graph in order to infer its hierarchical structure, i.e. binary tree.

The learned tree provides an induction of the senses of a given word at different levels of sense granularity and was applied to the problem of WSD. The WSD process mapped the tree's internal nodes to GS senses using a sense tagged corpus, and then tagged new instances by exploiting the structural information provided by the tree.

Our experimental results have shown that our graphs exhibit hierarchical organisation that can be captured by *HRGs*, in effect providing improved WSD performance compared to flat clustering. Additionally, our comparison against hierarchical agglomerative clustering with average-linkage has shown that *HRGs* perform significantly better than *HAC* when the graphs do not suffer from sparsity (disconnected graphs). The comparison with state-of-the-art sense induction systems has shown that our method yields improvements.

Our future work focuses on using different feature types, e.g. dependency relations, second-order co-occurrences, named entities and others to construct our undirected graphs and then applying HRGs, in order to measure the impact of each feature type on the induced hierarchical structures within a WSD setting. Moreover, following the work in (Clauset et al., 2008), we are also working on using MCMC in order to sample more than one dendrogram at equilibrium, and then combine them to a consensus tree. This consensus tree might be able to express a larger amount of topological features of the initial undirected graph.

Finally in terms of evaluation, our future work also focuses on evaluating *HRGs* using a fine-grained sense inventory, extending the evaluation on the SemEval-2010 WSI task dataset (Manandhar et al., 2010) as well as applying HRGs to other related tasks such as taxonomy learning.

Acknowledgements

This work is supported by the European Commission via the EU FP7 INDECT project, Grant No.218086, Research area: SEC-2007-1.2-01 Intelligent Urban Environment Observation System. The authors would like to thank the anonymous reviewers for their useful comments.

References

Eneko Agirre and Aitor. Soroa. 2007. Semeval-2007 Task 02: Evaluating Word Sense Induction and Discrimination Systems. In *Proceedings of SemEval-2007*, pages 7–12, Prague, Czech Republic.

Eneko Agirre, David Martínez, Oier López de Lacalle, and Aitor Soroa. 2006. Two Graph-based Algorithms for State-of-the-art WSD. In *Proceedings of EMNLP-2006*, pages 585–593, Sydney, Australia.

Chris Biemann. 2006. Chinese Whispers - An Efficient Graph Clustering Algorithm and its Application to Natural Language Processing Problems. In *Proceedings of TextGraphs*, pages 73–80, New York, USA.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.*, 3:993–1022.

Sergey Brin and Lawrence Page. 1998. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117.

Samuel Brody and Mirella Lapata. 2009. Bayesian Word Sense Induction. In *Proceedings of EACL-2009*, pages 103–111, Athens, Greece. ACL.

Aaron Clauset, Cristopher Moore, and Mark E. J. Newman. 2006. Structural Inference of Hierarchies in Networks. In *Proceedings of the ICML-2006 Workshop on Social Network Analysis*, pages 1–13, Pittsburgh, USA.

Aaron Clauset, Cristopher Moore, and Mark E. J. Newman. 2008. Hierarchical Structure and the Prediction of Missing Links in Networks. *Nature*, 453(7191):98–101.

Stijn Dongen. 2000. Performance Criteria for Graph Clustering and Markov Cluster Experiments. Technical report, CWI (Centre for Mathematics and Computer Science), Amsterdam, The Netherlands.

Beate Dorow and Dominic Widdows. 2003. Discovering Corpus-specific Word Senses. In *Proceedings of the EACL-2003*, pages 79–82, Budapest, Hungary.

Ted Dunning. 1993. Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics*, 19(1):61–74.

Phil Edmonds and Beate Dorow. 2001. Senseval-2: Overview. In *Proceedings of SensEval-2*, pages 1–5, Toulouse, France.

Ioannis P. Klapaftis and Suresh Manandhar. 2008. Word Sense Induction Using Graphs of Collocations. In *Proceedings of ECAI-2008*, pages 298–302, Patras, Greece.

Ioannis P. Klapaftis and Suresh Manandhar. 2010. Taxonomy Learning Using Word Sense Induction. In *Proceedings of NAACL-HLT-2010*, pages 82–90, Los Angeles, California, June. ACL.

Suresh Manandhar, Ioannis P. Klapaftis, Dmitriy Dligach, and Sameer S. Pradhan. 2010. Semeval-2010 Task 14: Word Sense Induction & Disambiguation. In *Proceedings of SemEval-2*, Uppsala, Sweden. ACL.

Rada Mihalcea. 2004. Graph-based Ranking Algorithms for Sentence Extraction, Applied to Text Summarization. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 20, Morristown, NJ, USA.

Mark Newman and Gerard Barkema. 1999. *Monte Carlo Methods in Statistical Physics*. Oxford: Clarendon Press, New York, USA.

Zheng-Yu Niu, Dong-Hong Ji, and Chew-Lim Tan. 2007. I2R: Three Systems for Word Sense Discrimination, Chinese Word Sense Disambiguation, and English Word Sense Disambiguation. In *Proceedings of SemEval-2007*, pages 177–182, Prague, Czech Republic.

Patrick Pantel and Dekang Lin. 2003. Automatically Discovering Word Senses. In *Proceedings of NAACL-HLT-2003*, pages 21–22, Morristown, NJ, USA.

Ted Pedersen and Anagha Kulkarni. 2006. Automatic Cluster Stopping With Criterion Functions and the gap Statistic. In *Proceedings of the 2006 Conference of the North American Chapter of the ACL on Human Language Technology*, pages 276–279, Morristown, NJ, USA.

Ted Pedersen. 2007. UMND2 : Senseclusters Applied to the Sense Induction Task of Senseval-4. In *Proceedings of SemEval-2007*, pages 394–397, Prague, Czech Republic.

Daniel Ramage, Anna N. Rafferty, and Christopher D. Manning. 2009. Random Walks for Text Semantic Similarity. In *Proceedings of TextGraphs-4*, Suntec, Singapore, August.

Noam Slonim, Nir Friedman, and Naftali Tishby. 2002. Unsupervised Document Classification Using Sequential Information Maximization. In *SIGIR 2002*, pages 129–136, New York, NY, USA. ACM.

Benjamin Snyder and Martha Palmer. 2004. The English All-words Task. In Rada Mihalcea and Phil Edmonds, editors, *In Proceedings of Senseval-3*, pages 41–43, Barcelona, Spain.

Jean Véronis. 2004. Hyperlex: Lexical Cartography for Information Retrieval. *Computer Speech & Language*, 18(3):223–252.

Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising Measures of Lexical Distributional Similarity. In *Proceedings of COLING-2004*, pages 10–15, Morristown, NJ, USA.

Dominic Widdows and Beate Dorow. 2002. A Graph Model for Unsupervised Lexical Acquisition. In *Proceedings of Coling-2002*, pages 1–7, Morristown, NJ, USA.

Towards Conversation Entailment: An Empirical Investigation

Chen Zhang **Joyce Y. Chai**
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824, USA
{zhangch6, jchai}@cse.msu.edu

Abstract

While a significant amount of research has been devoted to textual entailment, automated entailment from conversational scripts has received less attention. To address this limitation, this paper investigates the problem of conversation entailment: automated inference of hypotheses from conversation scripts. We examine two levels of semantic representations: a basic representation based on syntactic parsing from conversation utterances and an augmented representation taking into consideration of conversation structures. For each of these levels, we further explore two ways of capturing long distance relations between language constituents: implicit modeling based on the length of distance and explicit modeling based on actual patterns of relations. Our empirical findings have shown that the augmented representation with conversation structures is important, which achieves the best performance when combined with explicit modeling of long distance relations.

1 Introduction

Textual entailment has received increasing attention in recent years (Dagan et al., 2005; Bar-Haim et al., 2006; Giampiccolo et al., 2007; Giampiccolo et al., 2008; Bentivogli et al., 2009). Given a segment from a textual document, the task of textual entailment is to automatically determine whether a given hypothesis can be entailed from the segment. The capability of such kind of inference can benefit many text-based applications such as information extraction and question answering.

Textual entailment has mainly focused on inference from written text in monologue. Recent years also observed an increasing amount of conversational data such as conversation scripts of meetings, call center records, court proceedings, as well as online chatting. Although conversation is a form of language, it is different from monologue text with several unique characteristics. The key distinctive features include turn-taking between participants, grounding between participants, different linguistic phenomena of utterances, and conversation implicatures. Traditional approaches dealing with textual entailment were not designed to handle these unique conversation behaviors and thus to support automated entailment from conversation scripts.

Example 1:
Conversation Segment:
B: My mother also was very very independent. She had her own, still had her own little house and still driving her own car,
A: Yeah.
B: at age eighty-three.
Hypothesis:
(1) B's mother is eighty-three.
(2) B is eighty-three.

To address this limitation, our previous work (Zhang and Chai, 2009) has initiated an investigation on the problem of *conversation entailment*. The problem was formulated as follows: given a conversation discourse *D* and a hypothesis *H* concerning its participant, the goal was to identify whether *D* entails *H*. For instance, as in Example 1, the first hypothesis can be entailed from the

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 756–766,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

conversation segment while the second hypothesis cannot. While our previous work has provided some interesting preliminary observations, it mostly focused on data collection and initial experiments and analysis using a small set of development data. It is not clear whether the previous results are generally applicable, how different components in the entailment framework interact with each other, and how different representations may influence the entailment outcome.

To reach a better understanding of conversation entailment, we conducted a further investigation based on the larger set of test data collected in our previous work (Zhang and Chai, 2009). We specifically examined two levels of representations: a basic representation based on syntactic parsing from conversation utterances and an augmented representation taking into consideration of conversation structures. For each of these levels, we further explored two ways of capturing long distance relations: (1) implicit modeling based on the length of distance and (2) explicit modeling based on actual patterns of relations. Our empirical findings have shown that augmented representation with conversation structures is important in conversation entailment. Combining conversation structures with explicit modeling of long distance relations results in the best performance.

2 Related Work

Our work here is related to recent advances in textual entailment, automated processing of conversation scripts, and our initial investigation on conversation entailment.

There is a large body of work on textual entailment initiated by the Pascal Recognizing Textual Entailment (RTE) Challenges (Dagan et al., 2005; Bar-Haim et al., 2006; Giampiccolo et al., 2007; Giampiccolo et al., 2008; Bentivogli et al., 2009). Different approaches have been developed, for example, based on logic proving (Tatu and Moldovan, 2005; Bos and Markert, 2005; Raina et al., 2005) and graph match (Haghighi et al., 2005; de Salvo Braz et al., 2005; MacCartney et al., 2006). Supervised learning approaches have also been applied to measure the similarities between training and testing pairs (Zanzotto and Moschitti, 2006). In the most recent RTE Challenge (Bentivogli et al., 2009), the best system achieves 73.5% of accuracy, while the median performance among all participants is 60.4%. These results indicate that, while progress has been made, textual entailment remains a challenging problem.

As more and more conversation data becomes available, researchers have investigated automated processing of conversation data to acquire useful information, for example, related to opinions (Somasundaran et al., 2007; Somasundaran et al., 2008; Somasundaran et al., 2009), biographic attributes (Garera and Yarowsky, 2009), social networks (Jing et al., 2007), and agreements and disagreements between participants (Galley et al., 2004). Recent studies have also developed approaches to summarize conversations (Murray and Carenini, 2008) and to model conversation structures (dialogue acts) from online Twitter conversations (Ritter et al., 2010). Here we address a different angle regarding conversation scripts, namely conversation entailment.

In our previous work (Zhang and Chai, 2009), we started an initial investigation on conversation entailment. We have collected a dataset of 875 instances. Each instance consists of a conversation segment and a hypothesis (as described in Section 1). The hypotheses are statements about conversation participants and are further categorized into four types: about their profile information, their beliefs and opinions, their desires, and their communicative intentions. We developed an approach that is motivated by previous work on textual entailment. We use clauses in the logic-based approaches as the underlying representation of our system. Based on this representation, we apply a two stage entailment process similar to MacCartney et al. (2006) developed for textual entailment: an alignment stage followed by an entailment stage.

Building upon our previous work, in this paper, we systematically examine different representations of the conversation segment and different modeling of long distance relations between language constituents. We compare the roles of these different representations on the performance of entailment prediction using a larger testing dataset that was not previously evaluated. This analysis allows better understanding of the problem and provides insight on potential solutions.

3 Overall Framework

In our previous work (Zhang and Chai, 2009), conversation entailment is formulated as the following: given a conversation segment D which is represented by a set of clauses $D = d_1 \wedge \ldots \wedge d_m$, and a hypothesis H represented by another set of clauses $H = h_1 \wedge \ldots \wedge h_n$, the prediction on whether D entails H is determined by the product of probabilities that each hypothesis clause h_j is entailed from all the conversation segment clauses $d_1 \ldots d_m$ as follows. This is based on a simple assumption that whether a clause is entailed from a conversation segment is conditionally independent from other clauses.

$$\begin{aligned} & P(D \vDash H | D, H) \\ & \quad = P(D \vDash h_1, \ldots, D \vDash h_n | D, h_1, \ldots, h_n) \\ & \quad = \prod_{j=1}^{n} P(D \vDash h_j | D = d_1 \ldots d_m, h_j) \\ & \quad = \prod_{j=1}^{n} P(d_1 \ldots d_m \vDash h_j | d_1, \ldots, d_m, h_j) \qquad (1) \end{aligned}$$

A clause here is similar to a sentence in first-order predicate calculus. It is made up by *terms* and *predicates*. A term is either: 1) an entity described by a noun phrase, e.g., *John Lennon*, *mother*, or *she*; or 2) an action or event described by a verb phrase, e.g., *marry* in "John married Eva in 1940". A predicate represents either: 1) a property (i.e., unary) for a term, e.g., $Russian(company)$, or $recently(visit)$; or 2) a relation (i.e., binary) between two terms, e.g., $subj(visit, Prime\ Minister)$ and $obj(visit, Brazil)$ in "Prime Minister recently visited Brazil".

Given the clause representation, we follow the idea similar to MacCartney et al. (2006), and predict the entailment decision in two stages of processing: (1) an alignment model aligns terms in the hypothesis to terms in the conversation segment; and (2) an inference model predicts the entailment based on the alignment between the hypothesis and the conversation segment.

3.1 Alignment Model

An **alignment** is defined as a mapping function g between a term x in the conversation segment and a term y in the hypothesis. $g(x, y) = 1$ if x and y are aligned; otherwise $g(x, y) = 0$. It is possible that multiple terms from the segment are mapped to one term in the hypothesis ($g(x_1, y) = g(x_2, y) = 1$), or vice versa ($g(x, y_1) = g(x, y_2) = 1$). To predict these alignments, the problem is formulated as binary classification: given any two terms x from the conversation and y from the hypothesis, decide the value of their alignment function $g(x, y)$.

3.2 Inference Model

Once an alignment between a hypothesis and a conversation segment is established, an inference model is applied to predict whether the conversation segment entails the hypothesis given such alignment. More specifically, as shown in Equation 1, given a clause from the hypothesis h_j, a set of clauses from the conversation segment $d_1, \ldots, d_m$, and an alignment g between them, the goal is to predict whether $d_1, \ldots, d_m$ entails h_j under the alignment g.

The prediction is treated differently according to different types of clauses. If h_j is a property clause (i.e., takes one argument $h_j(\cdot)$), a property inference model is applied; otherwise (i.e., relational clauses with two arguments $h_j(\cdot, \cdot)$), a relational inference model is applied.

In this paper we follow the same framework. However our focus here is on the new question that how different levels of semantic representation and different approaches of modeling long distance relationship affect the alignment and inference models as well as the overall entailment performance.

4 Semantic Representation

Given the clause representation described earlier, an important question is what information from the conversation segment should be captured and represented. To address this question, we examined two levels of shallow semantic representation. The first level is basic representation which only captures the information from all the utterances in the conversation segment. The second representation includes conversation structures (e.g., speakers and dialogue acts). Next we use Example 2 to illustrate these representations.

Example 2:
Conversation Segment:
B: Have you seen *Sleeping with the Enemy*?
A: No. I've heard that's really great, though.
B: You have to go see that one.
Hypothesis:
B suggests A to watch *Sleeping with the Enemy*.

4.1 Basic Representation

The first representation is based on the syntactic parsing from conversation utterances and we call it a *basic* representation. Figure 1(a) shows an example of dependency structures for several utterances that are derived from the Stanford parser (Klein and Manning, 2003), and Figure 1(b) shows the corresponding clause representation. In the dependency structure, the vertices represent entities (e.g., x_1) and actions (e.g., x_3) within an utterance. They correspond to terms in the clause representation. An edge between vertices captures a dependency relation and is represented as predicates in the clause representation. For example, the edge between x_1 and x_3 indicates x_1 is the subject of x_3, which is represented by the clause representation $subj(x_3, x_1)$. Similar representation also applies to the hypothesis as shown in Figure 1(c), 1(d).

4.2 Augmented Representation

The second representation is built upon the basic representation and incorporates conversation structure across turns and utterances. We call it an *augmented* representation. Figure 2(a) shows the augmented structures of the conversation segment and Figure 2(b) shows the corresponding clause representation. Compared to the basic representation, there are two additional types of vertices (i.e., terms) highlighted in the figures:

- Vertices representing utterances (e.g., $u_1 \ldots u_4$). Their corresponding terms capture the dialogue acts for the utterances (e.g. $u_1 = yes\_no\_question$). To focus our effort, currently we only apply annotated dialogue acts provided in the Switchboard corpus (Godfrey and Holliman, 1997). Two edges are added to connect different utterances. The first edge connects each utterance vertex to the head of the corresponding utterance to indicate the specific content of the utterance (e.g., $content(u_1, x_3)$). The second edge connects an utterance to its succeeding utterance to indicate the temporal progression of the conversation (e.g., $follow(u_2, u_1)$).
- Vertices representing speakers or participants (e.g., s_A, s_B). One edge is added to connect each utterance to its speaker (e.g., $speaker(u_1, s_B)$).

Note that since our clause representations are mainly based on the dependency relations, they are mostly syntactic-driven. However, it does capture some shallow semantics such as who is the agent (i.e., subject) or the patient (i.e., object) of an event. The incorporation of speakers and dialogue acts in our augmented representations provides additional semantics of conversation discourse.

5 Modeling LDR

A critical part in predicting entailment is to recognize the semantic relationship between two language constituents, especially when these two constituents are not directly related. In Figure 2(a), for example, we want to recognize that x_9 (*You*) is the (logical) subject of x_{11} (see). Here we experimented two ways of modeling such long distance relations (LDR).

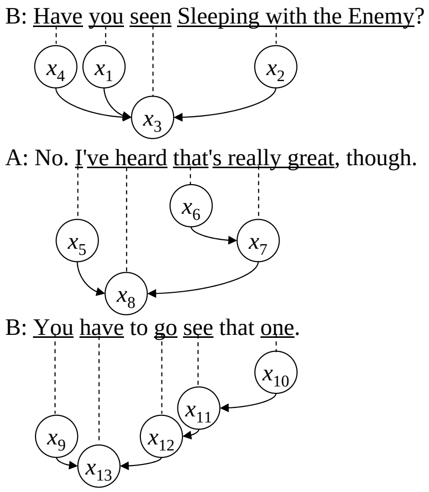
5.1 Implicit Modeling of LDR

The first method characterizes the relationship simply by the distance between two constituents in the dependency structure (or augmented structure). For example, in Figure 2(a) the distance between x_{11} and x_9 is 3. We call this method an *implicit* modeling of long distance relationship.

The advantage of implicit modeling is that it is easy to implement based on the dependency structure. However, its limitation is that the distance measure does not capture sufficient information of semantic relations between language constituents.

5.2 Explicit Modeling of LDR

The second way of modeling long distance relationship is called *explicit* modeling. It uses a string to


B: Have you seen Sleeping with the Enemy?
x_4
x_1
x_2
x_3
A: No. I've heard that's really great, though.
x_6
x_5
x_7
x_8
B: You have to go see that one.
x_{10}
x_{11}
x_9
x_{12}
x_{13}

(a) dependency structure of the conversation utterances

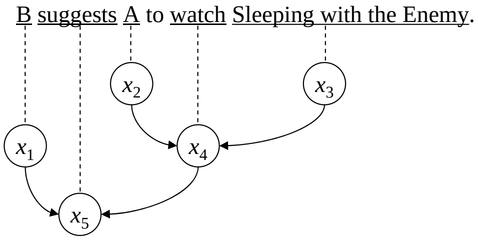
Terms	Clauses

x_1=A
x_2=Sleeping with the Enemy
x_3=seen, x_4=have
	obj(x_3,x_2)
	subj(x_3,x_1)
	aux(x_3,x_4)

x_5=A, x_6=that
x_7=is really great
x_8=have heard
	subj(x_7,x_6)
	obj(x_8,x_7)
	subj(x_8,x_5)

x_9=A, x_{10}=one,
x_{11}=see, x_{12}=go,
x_{13}=have
	obj(x_{11},x_{10})
	obj(x_{12},x_{11})
	obj(x_{13},x_{12})
	subj(x_{13},x_9)

(b) basic representation of the conversation segment



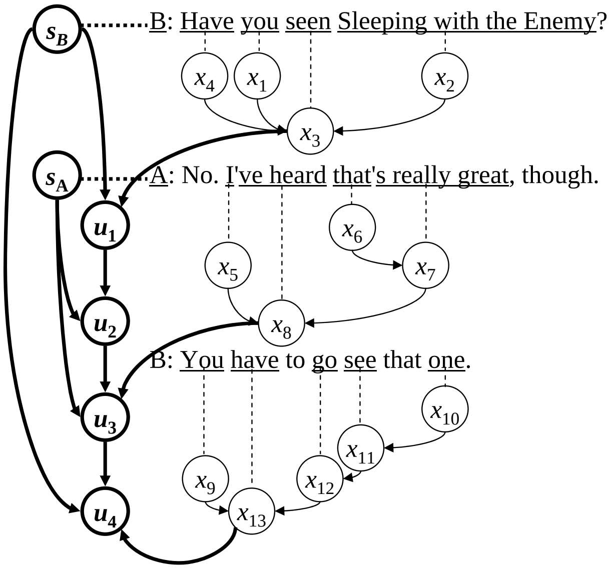
(c) dependency structure of the hypothesis

Terms	Clauses

x_1=B, x_2=A
x_3=Sleeping with the Enemy
x_4=watch
x_5=suggests
	subj(x_4,x_2)
	obj(x_4,x_3)
	subj(x_5,x_1)
	obj(x_5,x_4)

(d) representation of the hypothesis

Figure 1: The dependency structures and corresponding basic representation of Example 2


s_B
B: Have you seen Sleeping with the Enemy?
x_4
x_1
x_2
x_3
s_A
A: No. I've heard that's really great, though.
u_1
x_6
x_5
x_7
u_2
x_8
B: You have to go see that one.
u_3
x_{10}
x_{11}
x_9
x_{12}
x_{13}
u_4

(a) dependency and conversation structures of the conversation segment

Terms	Clauses

s_A, s_B
u_1=yes_no_question
	speaker(u_1,s_B)
	content(u_1, x_3)

x_1=A, x_3=seen, x_4=have
x_2=Sleeping with the Enemy
	obj(x_3,x_2), subj(x_3,x_1)
	aux(x_3,x_4)

u_2=no_answer	***speaker(u_2,s_A)***
	follow(u_2,u_1)

u_3=statement	***speaker(u_3,s_A)***
	content(u_3.x_8)
	follow(u_3,u_2)

x_5=A, x_7=is really great
x_6=that, x_8=have heard
	subj(x_7,x_6), obj(x_8,x_7)
	subj(x_8,x_5)

u_4=viewpoint	***speaker(u_4,s_B)***
	content(u_4,x_{13})
	follow(u_4,u_3)

x_9=A, x_{10}=one, x_{11}=see
x_{12}=go, x_{13}=have
	obj(x_{11},x_{10}), obj(x_{12},x_{11})
	obj(x_{13},x_{12}), subj(x_{13},x_9)

(b) augmented representation of the conversation segment

Figure 2: The dependency and conversation structures and corresponding augmented representation of Example 2

describe the path from one constituent to the other: $v_1 e_1 \ldots v_{l-1} e_{l-1} v_l$, where $v_1, \ldots, v_l$ are the vertices on the path and $e_1, \ldots, e_{l-1}$ are the edges. Each v_i describes the type of the vertex in the dependency structure, which is either a noun (N), a verb (V), or an utterance (U). Each e_i describes whether the edge is forward (→) or backward (←). For example, in Figure 2(a), the path from x_{11} to x_9 is $V \rightarrow V \rightarrow V \leftarrow N$.

This kind of string representation of paths in syntactic parse is known as a way of modeling "shallow semantics" between any two constituents in a language structure. It is largely used in other NLP tasks such as semantic role labeling (Pradhan et al., 2008). The difference here is our paths are extracted from dependency parses as opposed to traditional constituent parses, and our paths also incorporate the representation of conversation structures (e.g., utterances and speakers).

6 Applications in Entailment Models

In this section we describe how different representations and modeling of LDR are used in the alignment and inference models.

6.1 Applications in Alignment Model

Although a noun and a verb can potentially be aligned, to simplify the problem, we restrict the problem to the alignment between two nouns or two verbs. We trained an alignment model for nouns and one for verbs separately.

Table 1 summarizes a set of features used in the alignment models. Most of these features are shared by the model for noun alignment and the model for verb alignment. These features include whether the two strings are the same, two terms have the same stem, the similarity between the two terms either based on WordNet or distributional statistics (Lin, 1998).

To learn the alignment model for nouns, we annotated the noun alignments for the development data used in PASCAL RTE-3 Challenge (Giampiccolo et al., 2007) and trained a logistic regression model based on the features in Table 1. Cross-validation on the same dataset shows relatively satisfying performance (96.4% precision and 94.9% recall). In this paper, we focus on the alignment between verbs since it appears more difficult.

	Noun Align.	Verb Align.
Verb *be* identification		✓
String equality	✓	✓
Stemmed equality	✓	✓
Acronym equality	✓	
Named entity equality	✓	
WordNet similarity	✓	✓
Distributional similarity	✓	✓
Subject consistency		✓
Object consistency		✓

Table 1: Features for alignment models

A major difference between noun alignment and verb alignment is that, for verb alignment the consistency of their arguments is also important. For two events (described by two verbs) to be aligned, at least their subjects (usually denoting the executers of actions) and objects (usually denoting the receivers of actions) should match to each other respectively. Note that, although actions/events also depend on other arguments or adjuncts, here we only consider the subjects and objects and leave the consistency check of other arguments/adjuncts to downstream processes. Based on two different ways of modeling long distance relationship (as described in Section 5), we explored two methods for modeling argument consistency (AC) in verb alignment models.

6.1.1 Implicit Modeling of AC

The first approach models argument consistency based on implicit modeling of the relationship between a verb and its aligned subject/object. Specifically, given a pair of verb terms (x, y) where x is from the conversation segment and y is from the hypothesis, let s_y be the subject of y and s_x be the aligned entity of s_y in the conversation (in case of multiple alignments, s_x is the one closest to x). The subject consistency of the verbs (x, y) is then measured by the distance between s_x and x in the dependency structure. Similarly, the distance between a verb and its aligned object is used as a measure of the object consistency.

In Example 2, to decide whether the conversation term *see* (x_{11} in Figure 1(a), 1(b), and 2) and the hypothesis term *watch* (x_4 in Figure 1(c), 1(d)) should be aligned, we first identify the subject of x_4 in the hypothesis, which is x_2 (A). We then look for

x_2's alignments in the conversation segment, among which x_9 (*You*) is the closest to x_{11} (*see*). In Figure 2(a), we find the distance between x_{11} and x_9 is 3.

Using the implicit modeling of argument consistency, we follow the same approach as in our previous work (Zhang and Chai, 2009) and trained a logistic regression model to predict verb alignment based on the features in Table 1.

6.1.2 Explicit Modeling of AC

The second approach captures argument consistency based on explicit modeling of the relationship between a verb and its aligned subject (or object). Given a pair of verb terms (x, y), let s_y be the subject of y and s_x be the aligned entity of s_y in the conversation closest to x, we use the string describing the path from x to s_x as the feature to capture subject consistency. For example, in Figure 2(a), the path from x_{11} to x_9 is $V \rightarrow V \rightarrow V \leftarrow N$.

This string representation of paths is used to capture both the subject consistency and the object consistency. Since they are non-numerical features, and the variability of their values can be extremely large, so we applied an instance-based classification model (e.g., k-nearest neighbor) to determine alignments between verb terms. We measure the distance between two path features by their minimal string edit distance, and then simply use the Euclidean distance to measure the closeness between any two verbs. Again this model is trained from our development data described in Zhang and Chai (2009).

Figure 3 shows an example of alignment between the conversation terms and hypothesis terms in Example 2. Note that in this figure the alignment between $x_5 = suggests$ from the hypothesis and $u_4 = opinion$ from the conversation segment is a *pseudo alignment*, which directly maps a verb term in the hypothesis to an utterance term represented by its dialogue act. This alignment is obtained by following the same set of rules learned from the development dataset as in (Zhang and Chai, 2009).

6.2 Applications in Inference Model

As mentioned earlier, once an alignment is established, the inference model is to predict whether each clause in the hypothesis is entailed from the conversation segment. Two separate models were used to handle the inference of property clauses ($h_j(x)$) and and the inference of relational clauses ($h_j(x, y)$). Property clauses involve less variables and are relatively simple, so we used the same property inference model as in (Zhang and Chai, 2009). Here we focus on relational inference model and examine how different modeling of long distance relationship may affect relation inference.

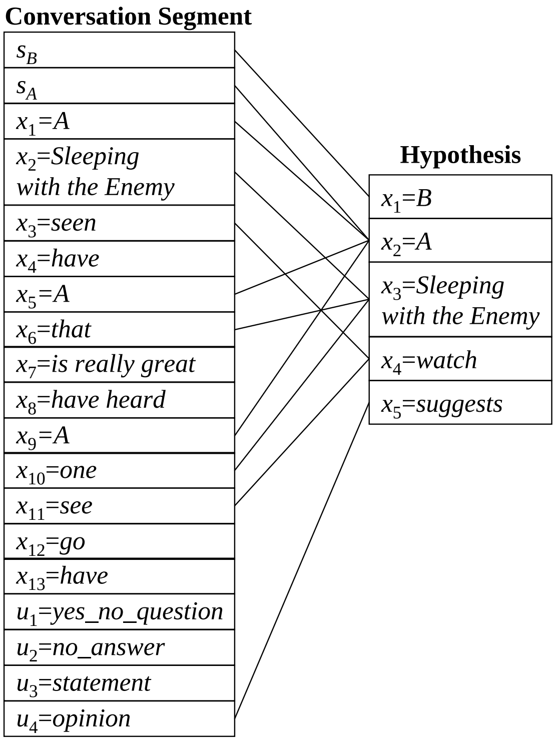


Figure 3: The alignment result for Example 2

For a relation h between x and y to be entailed from a conversation segment, we need to find a same or similar relation in the conversation segment between x's and y's counterparts (i.e., aligned entities of x and y in the conversation segment).

More specifically, given a relational clause from the hypothesis, $h_j(x, y)$, we find the sets of terms $X' = \{x' | x' \in D, g(x', x) = 1\}$ and $Y' = \{y' | y' \in D, g(y', y) = 1\}$, which are aligned with x and y, respectively. We then find the closest relation between these two sets of terms, (x^*, y^*), such that the distance between x^* and y^* is the smallest for any $x^* \in X'$ and $y^* \in Y'$. For instance, in the hypothesis of Example 2 there are terms x_5=*suggests* and x_4=*watch*, and a relational clause $obj(x_5, x_4)$ describing an action-object relation between them. Their counterparts in the conversation segment are $X' = \{u_4 = viewpoint\}$ and $Y' = \{x_3 = seen, x_{11} = see\}$. So the closest pair of terms between these two sets is u_4 and x_{11}. Consequently, whether the target relational clause $h_j(x, y)$ is entailed is determined by the relationship between x^* and y^*. Such relationship can be modeled either implicitly or explicitly.

6.3 Implicit modeling of relation inference

In this model we follow the simple idea that the shorter a path is between two terms, the more likely these two terms have a direct relationship. So we predefine a threshold, λ_L. We predict that $h_j(x, y)$ is entailed if the distance between x^* and y^* is smaller than λ_L. However, as can be seen, this distance does not reflect whether the type of relationship between x^* and y^* is similar to the relationship that holds between x and y.

6.4 Explicit modeling of relation inference

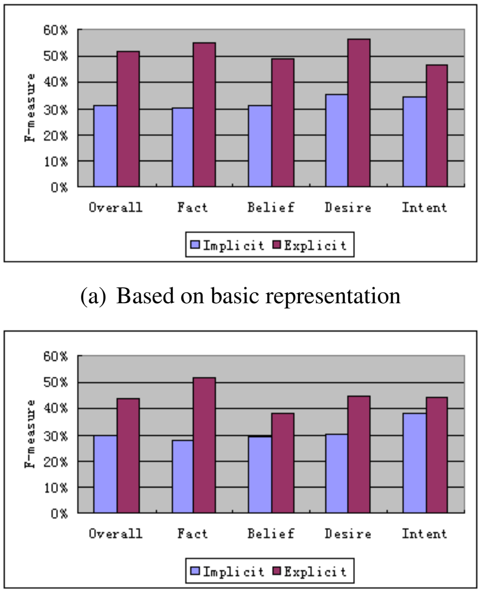
In order to capture more semantics from the relation between two terms, we use explicit modeling of the relationship between terms x^* and y^*. In the previous example, the relationship between u_4 and x_{11} is modeled by the path from u_4 to x_{11}, $U \leftarrow V \leftarrow V \leftarrow V$.

Given this characterization, the prediction of whether $h_j(x, y)$ is entailed from the conversation segment is formulated as a binary classification problem, using a k-nearest neighbor classification model with following features:

1. Explicit modeling of long distance relationship, i.e., the path from x^* to y^* in the dependency structure of the conversation segment;
2. The types (*N*, *V*, or *U*) of x, y, x^*, and y^*;
3. The type of relation between x and y, for example, *obj* in $obj(x, y)$;
4. The order (i.e., before or after) between x and y, and between x^* and y^*;
5. The specific type of the hypothesis.

7 Evaluation and Analysis

We evaluated different model configurations using our data[1]. This dataset consists of 291 development instances and 584 testing instances. The hypotheses were categorized into four types: (1) fact: profile and social relations of conversation participants (accounted for 47% of the development data and 49% of the testing data); (2) belief: participants' beliefs and opinions (34% and 35%); (3) desire: participants' desire of certain actions or outcomes (11% and 4%); (4) intent: communicative intent that captures some perlocutionary force from one participant to the other (e.g. A stops B from doing something; A disagreees with B on something, 8% and 12%)



(a) Based on basic representation

(b) Based on augmented representation

Figure 4: Evaluation of verb alignment

Note that in our original work (Zhang and Chai, 2009), only development data were used to show some initial observations. Here we trained our models on the development data and results shown are from the testing data.

7.1 Evaluation of Alignment Models

The evaluation of alignment models is based on pairwise decision. For each pair of terms (x, y), where x is from a conversation segment and y is from a hypothesis, we measure whether the model correctly predicts that the two terms should or should not be aligned. Because the alignment classification has extremely unbalanced classes, we use precision-recall of true alignments as evaluation metrics.

Figure 4(a) and 4(b) shows the comparison (F-measure) of two alignment models for verb align-

[1] The data is available for download at `http://links.cse.msu.edu:8000/lair/projects/conversationentailment_data.html`.

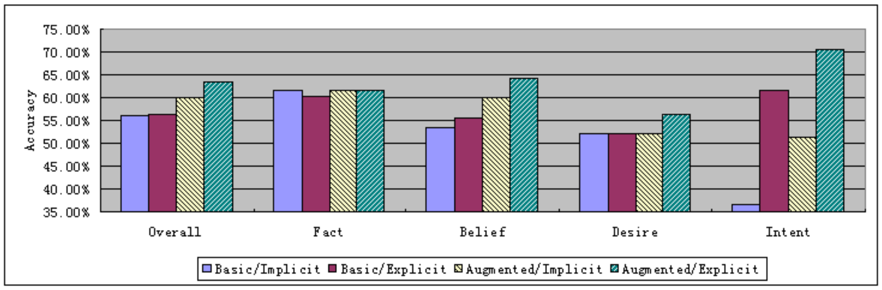


Figure 5: Evaluation of inference models based on different representations

ment, based on the basic representation and the augmented representation, respectively. Note that we cannot directly compare the results between these two figures since they involve different number of alignment instances[2]. Nevertheless, we can see the overall trend within each figure: the explicit model outperforms the implicit model. This suggests that the explicit modeling of semantic relationship between verbs and arguments works better than the implicit modeling used in previous work. Furthermore, the improvement is most noticeable when hypotheses are *facts* (24.8% with the basic representation and 24.1% with the augmented representation), and least when hypotheses are *intents* (12.2% with the basic representation and 6.2% with the augmented representation).

7.2 Evaluation of Inference Models

In order to compare different inference models, in this section (and this section only) we use gold-standard alignment results. They are obtained from manual annotation in our evaluation. We evaluated two inference models, one with implicit modeling of long distance relationship and one with explicit modeling. Evaluations were conducted based on both the basic representation and the augmented representation. Figure 5 shows the four groups of evaluation results.

Overall speaking, the augmented representation outperforms the basic representation for both implicit modeling and explicit modeling of long distance relationship (McNemar's tests, $p < 0.05$). The explicit model performs better than implicit model only based on augmented representation (McNemar's test, $p < 0.05$).

Clause Representation	Relation modeling		Improvement
	Implicit	Explicit	
Basic	53.9%	53.9%	0
Augmented	54.8%	58.7%	**3.9%**

Table 2: Entailment performance with different representations and LDR modeling

The results were further broken down by different hypothesis types. For the *fact* type of hypotheses, there is no difference between different representations and modeling of long distance relationship. This is not surprising since most hypotheses about partipants' profiling information can be inferred directly from the utterances. The augmented representation affects the *intent* type of hypothesis most significantly, so does the explicit modeling of long distance relationship.

7.3 Interaction between Clause Representations and LDR Modeling

It was shown in previous sections that the augmented representation helps entailment prediction compared to the basic representation. Here we want to study how they interact with other entailment components and what is their effect in the enhanced modeling of long distance relations. Specifically, we test the performance of implicit and explicit modeling of long distance relations under two different representation settings: the basic representation and the augmented representation.

Table 2 compares the performance (accuracy) of entailment models with different relationship modeling. We can see that the explicit model makes improvement over the implicit model for augmented representation (McNemar's test, $p < 0.05$), while no improvement is made for basic representation. These evaluation results appear to suggest that there

[2] The alignment based on the augmented representation in Figure 4(b) also includes pseudo alignments.

is an interaction between clause representations and semantic modeling of long distance relations: the modeling of long distance relations between language constituents appears only effective when conversation structure is incorporated in the representation.

It is interesting to see the difference in the prediction performances on *fact* hypotheses and *intent* hypotheses. For *fact*, the most benefit of incorporating explicit modeling of long distance relationship appears at the alignment stage, but not much at the inference stage. However, this situation is different for *intent*, where the benefit of explicitly modeling long distance relationship mostly happened at the inference stage. This observation suggests that the effects of different types of modeling may vary for different types of hypotheses, which indicates that hypothesis type dependent models may be beneficial.

8 Discussion and Conclusion

This paper presents an empirical investigation on conversation entailment. We specifically examine two levels of representation of conversation segments and two different ways of modeling long distance relations between language constituents. Our findings indicate that, although traditional architecture and approaches for textual entailment remain important, additional representation and processing that address conversation structures is critical. The augmented representation with conversation structures, together with explicit modeling of semantic relations between language constituents, results in the best performance (58.7% accuracy).

The work here only represents an initial step towards conversation entailment. Conversation phenomena are rich and complex. Conversation entailment is extremely difficult. Besides the same challenges faced by textual entailment, it is further complicated by conversation implicature. Although our current data enables us to start an initial investigation, its small size poses significant limitations on technology development and evaluation. For example, our studies have indicated hypothesis type-dependent approaches may be beneficial, however we do not have sufficient data to yield reasonable models. A more systematical approach to collect and create a larger set of data is crucial. Innovative community-based approaches (e.g., through web) for data collection and annotation can be pursued in the future. As more techniques in semantic processing (e.g., semantic role) become available, future work should also capture deeper semantics, address pragmatics, and incorporate richer world knowledge.

Finally, as the technology in conversation entailment is developed, its applications in NLP problems should be explored. Example applications include information extraction, question answering, summarization from conversation scripts, and modeling of conversation participants. These applications may provide new insights on the nature of the conversation entailment problem and its potential solutions.

Acknowledgments

This work was supported by grant IIS-0347548 from the National Science Foundation. We thank the anonymous reviewers for their valuable comments and suggestions.

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.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth pascal recognizing textual entailment challenge. In *Proceedings of the Second Text Analysis Conference (TAC 2009)*.

Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proceedings of HLT-EMNLP*, pages 628–635.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *PASCAL Challenges Workshop on Recognising Textual Entailment*.

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 AAAI*.

Michel Galley, Kathleen McKeown, Julia Hirschberg, and Elizabeth Shriberg. 2004. Identifying agreement and disagreement in conversational speech: Use of bayesian networks to model pragmatic dependencies. In *Proceedings of ACL*, pages 669–676.

Nikesh Garera and David Yarowsky. 2009. Modeling latent biographic attributes in conversational genres. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 710–718, Suntec, Singapore, August.

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.

Danilo Giampiccolo, Hoa Trang Dang, Bernardog Magnini, Ido Dagan, Elena Cabrio, and Bill Dolan. 2008. The fourth pascal recognizing textual entailment challenge. In *Proceedings of the First Text Analysis Conference (TAC 2008)*.

John J. Godfrey and Edward Holliman. 1997. *Switchboard-1 Release 2*. Linguistic Data Consortium, Philadelphia.

Aria Haghighi, Andrew Ng, and Christopher Manning. 2005. Robust textual inference via graph matching. In *Proceedings of HLT-EMNLP*, pages 387–394.

Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2007. Extracting social networks and biographical facts from conversational speech transcripts. In *Proceedings of ACL*, pages 1040–1047.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 423–430, Morristown, NJ, USA.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of International Conference on Machine Learning*, pages 296–304.

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 HLT-NAACL*, pages 41–48.

Gabriel Murray and Giuseppe Carenini. 2008. Summarizing spoken and written conversations. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 773–782, Honolulu, Hawaii, October.

Sameer S. Pradhan, Wayne Ward, and James H. Martin. 2008. Towards robust semantic role labeling. *Computational Linguistics*, 34(2):289–310.

Rajat Raina, Andrew Y. Ng, and Christopher D. Manning. 2005. Robust textual inference via learning and abductive reasoning. In *Proceedings of AAAI*, pages 1099–1105.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 172–180, Los Angeles, California, June.

Swapna Somasundaran, Josef Ruppenhofer, and Janyce Wiebe. 2007. Detecting arguing and sentiment in meetings. In *Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue*, Antwerp, September.

Swapna Somasundaran, Janyce Wiebe, and Josef Ruppenhofer. 2008. Discourse level opinion interpretation. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 801–808, Manchester, UK, August.

Swapna Somasundaran, Galileo Namata, Janyce Wiebe, and Lise Getoor. 2009. Supervised and unsupervised methods in employing discourse relations for improving opinion polarity classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 170–179, Singapore, August.

Marta Tatu and Dan Moldovan. 2005. A semantic approach to recognizing textual entailment. In *Proceedings of HLT-EMNLP*, pages 371–378.

Fabio Massimo Zanzotto and Alessandro Moschitti. 2006. Automatic learning of textual entailments with cross-pair similarities. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 401–408, Morristown, NJ, USA.

Chen Zhang and Joyce Chai. 2009. What do we know about conversation participants: Experiments on conversation entailment. In *Proceedings of the SIGDIAL 2009 Conference*, pages 206–215.

The Necessity of Combining Adaptation Methods

Ming-Wei Chang, Michael Connor and Dan Roth
University of Illinois at Urbana Champaign
Urbana, IL 61801
{mchang21,connor2,danr}@uiuc.edu

Abstract

Problems stemming from domain adaptation continue to plague the statistical natural language processing community. There has been continuing work trying to find general purpose algorithms to alleviate this problem. In this paper we argue that existing general purpose approaches usually only focus on one of two issues related to the difficulties faced by adaptation: 1) difference in base feature statistics or 2) task differences that can be detected with labeled data.

We argue that it is *necessary* to combine these two classes of adaptation algorithms, using evidence collected through theoretical analysis and simulated and real-world data experiments. We find that the combined approach often outperforms the individual adaptation approaches. By combining simple approaches from each class of adaptation algorithm, we achieve state-of-the-art results for both Named Entity Recognition adaptation task and the Preposition Sense Disambiguation adaptation task. Second, we also show that applying an adaptation algorithm that finds shared representation between domains often impacts the choice in adaptation algorithm that makes use of target labeled data.

1 Introduction

While recent advances in statistical modeling for natural language processing are exciting, the problem of domain adaptation remains a big challenge. It is widely known that a classifier trained on one domain (e.g. news domain) usually performs poorly on a different domain (e.g. medical domain) (Jiang and Zhai, 2007; Daumé III, 2007). The inability of current statistical models to handle multiple domains is one of the key obstacles hindering the progress of NLP.

Several general purpose algorithms have been proposed to address the domain adaptation problem: (Blitzer et al., 2006; Jiang and Zhai, 2007; Daumé III, 2007; Finkel and Manning, 2009). It is widely believed that the drop in performance of statistical models on new domains is due to the shift of the joint distribution of labels and examples, $P(Y, X)$, from domain to domain, where X represents the input space and Y represents the output space. In general, we can separate existing adaptation algorithms into two categories:

Focuses on $P(X)$ This type of adaptation algorithm attempts to resolve the difference between the feature space statistics of two domains. While many different techniques have been proposed, the common goal of these algorithms is to find a better shared representation that brings the source domain and the target domain closer. Often these algorithms do not use labeled examples in the target domain. The works (Blitzer et al., 2006; Huang and Yates, 2009) all belong to this category.

Focuses on $P(Y|X)$ These adaptation algorithms assume that there exists a small amount of labeled data for the target domain. Instead of training two weight vectors independently (one for source and the other for the target domain), these algorithms try to relate the source and target weight vectors. This is often achieved by using a special designed regularization term. The works (Chelba and Acero, 2004; Daumé III, 2007; Finkel and Manning, 2009) belong to this category.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 767–777,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

It is important to give the definition of an adaptation *framework*. An adaptation framework is specified by the data/resources used and a specific learning algorithm. For example, a framework that used only source labeled examples and one that used both source and target labeled examples should be considered as two different frameworks, even though they might use exactly the same training algorithm. Note that the goal of a good adaptation framework is to perform well on the target domain and quite often we only need to change the data/resource used to increase the performance without changing the training algorithm. We refer to frameworks that do not use target labeled data and focus on $P(X)$ as **Unlabeled Adaptation Frameworks** and refer to frameworks that use algorithms that focus on $P(Y|X)$ as **Labeled Adaptation Frameworks**.

The major difference between unlabeled adaptation frameworks and labeled adaptation frameworks is the use of *target labeled examples*. Unlabeled adaptation frameworks do not use target labeled examples[1], while the labeled adaptation frameworks make use of *target labeled examples*. Under this definition, we consider that a model trained on both source and target labeled examples (later referred as **S**+**T**) is a labeled adaptation framework.

It is important to combine the labeled and unlabeled adaptation frameworks for two reasons:

- **Mutual Benefit**: We analyze these two types of frameworks and find that they address different adaptation issues. Therefore, it is often beneficial to apply them together.

- **Complex Interaction**: Another, probably more important issue, is that these two types of frameworks are *not* independent. Different representations will impact how much a labeled adaptation algorithm can transfer information between domains. Therefore, in order to have a clear picture of what is the best labeled adaptation framework, it is necessary to analyze these two domain adaptation frameworks together.

In this paper, we assume we have both a small amount of target labeled data and a large amount of unlabeled data so that we can perform both unlabeled and labeled adaptation. *The goal of our paper is to point out the* **necessity** *of applying these two adaptation frameworks together*. To the best of our knowledge, this is the first paper that both theoretically and empirically analyzes the interdependence between the impact of labeled and unlabeled adaptation frameworks.

The contribution of this paper is as follows:

- Propose a theoretical analysis of the "Frustratingly Easy" (FE) framework (Daumé III, 2007) (Section 3).

  The theoretical analysis shows that for FE to be effective the domains must already be "close". At some threshold of "closeness" it is better to switch from FE to just pool all training together as one domain.

- Demonstrate the complex interaction between unlabeled and labeled approaches (Section 4)

  We construct artificial experiments that demonstrate how applying unlabeled adaptation may impact the behavior of two labeled adaptation approaches.

- Empirically analyze the interaction on real datasets (Section 5).

  We show that in general combining both approaches on the tasks of preposition sense disambiguation and named entity recognition works better than either individual method. Our approach not only achieves state-of-the-art results on these two tasks but it also reveals something surprising – finding a better shared representation often makes a simple source+target approach the best adaptation framework in practice.

2 Two Adaptation Aspects: A Review

Why do we need two types of adaptation frameworks? First, unlabeled adaptation frameworks are necessary since many features only exist in one domain. Therefore, it is important to develop algorithms that find features which work across domains. On the other hand, labeled adaptation frameworks

[1] Note that we still use labeled data from source domain in an unlabeled adaptation framework.

are also required because we would like to take advantages of target labeled data. Even though different domains may have different definitions for labels (say in named entity recognition, specific definition of PER/LOC/ORG may change), labeled data should still be useful. We summarize these distinctions in Table 1.

While these two aspects of adaptation both saw significant progress in the past few years, little analysis has been done on the interaction between these two types of algorithms[2].

In order to have a deep analysis, it is necessary to choose specific adaptation algorithms for each aspect of adaptation framework. While we mainly conduct analysis on the algorithms we picked, we would like to point out that the necessity of combining these two types of adaptation algorithms has been largely ignored in the community.

As our example adaptation algorithms we selected:

Labeled adaptation: FE framework One of the most popular adaptation frameworks that requires the use of labeled target data is the "Frustratingly Easy" (FE) adaptation framework (Daumé III, 2007). However, why and when this framework works remains unclear in the NLP community. The FE framework can be viewed as an framework that extends the feature space, and it requires source and target labeled data to work. We denote n as the total number of features[3] and m is the number of the "domains", where one of the domains is the target domain. The FE framework creates a global weight vector in $\mathbb{R}^{n(m+1)}$, an extended space for all domains. The representation $\mathbf{x}$ of the t-th domain is mapped by $\Phi_t(\mathbf{x}) \in \mathbb{R}^{n(m+1)}$. In the extended space, the first n features consist of the "shared" block, which is always active across all tasks. The $(t+1)$-th block (the $(nt+1)$-th to the $(nt+n)$-th features) is a "specific" block, and is only active when extracting examples from the task t. More formally,

$$\Phi_t(\mathbf{x}) = \begin{bmatrix} \underbrace{\mathbf{x}}_{\text{shared}} & \overbrace{\mathbf{0}\ldots\mathbf{0}}^{(t-1)\text{ blocks}} & \underbrace{\mathbf{x}}_{\text{specific}} & \overbrace{\mathbf{0}\ldots\mathbf{0}}^{(m-t)\text{ blocks}} \end{bmatrix}. \tag{1}$$

A *single* weight vector $\bar{\mathbf{w}}$ is obtained by training on the modified labeled data $\{y_i^t, \Phi_t(\mathbf{x}_i^t)\}_{t=1}^m$. Given that this framework only extends the feature space, in this paper, we also call it the *feature extension* framework (still called FE). We will see in Section 3 that this framework is equivalent to applying a regularization trick that bridges the source and the target domains. As it will become clear in Section 3, in fact, this framework is only effective when there is target labeled data and hence belongs to labeled adaptation frameworks.

Although FE framework is quite popular in the community, there are other even simpler labeled adaptation frameworks that allow the use of target labeled data. For example, one of the simplest frameworks is the **S+T** framework, which simply trains a single model on the pooled and unextended source and target training data.

Unlabeled adaptation: Adding cluster-like features Recall that unlabeled adaptation frameworks find the features that "work" across domain. In this paper, we find such features in two steps. First, we use word clusters generated from unlabeled text and/or third party resources that spans domains. Then, for every feature template that contains a word, we *append* another feature template that uses the word's cluster instead of the word itself. This technique is used in many recent works including dependency parsing and NER (Koo et al., 2008; Ratinov and Roth, 2009). Note that the unlabeled text need not come from the source or target domain. In fact, in this paper, we use clusters generated with the Reuters 1996 dataset, a superset of the CoNLL03 NER dataset (Koo et al., 2008; Liang, 2005). We adopt the Brown cluster algorithm to find the word cluster (Brown et al., 1992; Liang, 2005). We can use other resources to create clusters as well. For example, in the NER domain, we also include gazetteers[4] as an unlabeled cluster resource, which can bring the domains together quite effectively.

[2] Among the previously mentioned work, (Jiang and Zhai, 2007) is a special case given that it discusses both aspects of adaptation algorithms. However, little analysis on the interaction of the two aspects is discussed in that paper

[3] We assume that the number of features in each domain is equal.

[4] Our gazetteers comes from (Ratinov and Roth, 2009).

Framework	Labeled Data	Unlabeled Data	Common Approach
Unlabeled Adaptation (Focus on $P(X)$)	Source	Encompasses Source and Target. May use other third party resources (dictionaries, gazetteers, etc.).	Generate features that span domains using unlabeled data and/or third party resources.
Labeled Adaptation (Focus on $P(Y\|X)$)	Source and Target	None	Train classifier(s) using both source and target training data, relating the two.

Table 1: Comparison between two general adaptation frameworks discussed in this paper. Each framework is specified by its setting (data required) and its learning algorithm. Multiple previous adaptation approaches fit in one of either framework.

While other more complex algorithms (Ando and Zhang, 2005; Blitzer et al., 2006) for finding better shared representation (without using labeled target data) have been proposed, we find that using straightforward clustering features is quite effective in general.

3 Analysis of the FE Framework

In this section, we propose a simple yet informative analysis of the FE algorithm from the perspective of multi-task learning. *Note that we ignore the effect of unlabeled adaptation in this section, and focus on the analysis of the FE framework as a representative labeled adaptation framework.*

3.1 Mistake Bound Analysis

While (Daumé III, 2007) proposed this framework for adaptation, a very similar idea had been proposed in (Evgeniou and Pontil, 2004) as a novel regularization term for *multitask* learning with support vector machines. Assume that $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m$ are the weight vector for the first domain to the m-th domain, respectively. The baseline approach is to assume that each weight vector is independent. Assume that we adopt a SVM-like optimization problem that consider all m tasks, the baseline approach is equivalent to using the following regularization term in the objective function: $\sum_{t=1}^{m} \|\mathbf{w}_t\|^2$.

In (Evgeniou and Pontil, 2004; Daumé III, 2007), they assume that $\mathbf{w}_t = \mathbf{u} + \mathbf{v}_t$, for $t = 1, \ldots m$, where $\mathbf{v}_t$ is the specific weight vector for t-th domain and $\mathbf{u}$ is a *shared* weight vector across all domains. The new regularization term then becomes

$$\|\mathbf{u}\|^2 + \sum_{t=1}^{m} \|\mathbf{v}_t\|^2. \quad (2)$$

Note that these two regularization terms are different, given that the new regularization term makes $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m$ not independent anymore. It follows that

$$\mathbf{w}_t^T \mathbf{x} = (\mathbf{u} + \mathbf{v}_t)^T \mathbf{x} = \bar{\mathbf{w}}^T \Phi_t(\mathbf{x}),$$

where

$$\bar{\mathbf{w}}^T = \begin{bmatrix} \mathbf{u}^T & \mathbf{v}_1^T & \ldots & \mathbf{v}_m^T \end{bmatrix}.$$

and $\|\bar{\mathbf{w}}\|^2$ equals to Eq. (2). Therefore, we can think feature extension framework as a learning framework that adopts Eq. (2) as its regularization term.

The **FE** framework was in fact originally designed for the problem of multitask learning so in the following, we propose a simple mistake bound analysis based on the *multitask* setting, where we calculate the mistakes on *all* domains[5]. We focus on multitask setting for two reasons: 1) the analysis is very easy and intuitive, and 2) in Section 4.1, we empirically confirm that the analysis holds for the adaptation setting.

In the following, we assume that the training algorithm used in the FE framework is the online perceptron learning algorithm (Novikoff, 1963). This allows us to analyze the mistake bound of the **FE** framework with the perceptron algorithm. The bound can give us an insight on when and why one should adopt the **FE** framework. By using the standard mistake bound theorem (Novikoff, 1963), we show:

Theorem 1. *Let $\mathcal{D}_t$ be the labeled data of domain t. Assume that there exist $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m$ such that*

$$y\mathbf{w}_t^T \mathbf{x} \geq \mu, \forall (\mathbf{x}, y) \in \mathcal{D}_t,$$

and assume that $\max_{(\mathbf{x},y)\in\mathcal{D}_t} \|\mathbf{x}\| \leq R^2, \forall t = 1 \ldots m$. *Then, the number of mistakes made with online perceptron training (Novikoff, 1963) and the*

[5] In the adaptation setting, one generally only cares about the performance on the target domain.

FE framework is bounded by

$$\frac{2R^2}{\mu^2}(\sum_{t=1}^{m} \|\mathbf{w}_t\|^2 - \frac{\| \sum_{t=1}^{m} \mathbf{w}_t \|^2}{m+1}). \qquad (3)$$

Proof. Define $\bar{\mathbf{w}}$ as a vector in $\mathbb{R}^{n(m+1)}$. We claim that there exists a set $S_{\bar{\mathbf{w}}}$ such that for all $\bar{\mathbf{w}} \in S_{\bar{\mathbf{w}}}$, $\bar{\mathbf{w}}^T \Phi_t(\mathbf{x}) = \mathbf{w}_t^T \mathbf{x}$ for any domain $t = 1 \ldots m$. Note that $\Phi_t(x)$ is defined in Eq. (1). We can construct $S_{\bar{\mathbf{w}}}$ in the following way:

$$S_{\bar{\mathbf{w}}} = \{\begin{bmatrix} s & (\mathbf{w}_1 - s) & \ldots & (\mathbf{w}_m - s)\end{bmatrix} \mid s \in \mathbb{R}^n\},$$

where s is an arbitrary vector with n elements.

In order to obtain the best possible bound, we would like to find the most compressed weight vector in $S_{\bar{\mathbf{w}}}$, $\mathbf{w}^* = \min_{\bar{\mathbf{w}} \in S_{\bar{\mathbf{w}}}} \|\bar{\mathbf{w}}\|^2$.

The optimization problem has an analytical solution:

$$\|\mathbf{w}^*\|^2 = \sum_{t=1}^{m} \|\mathbf{w}_t\|^2 - \| \sum_{t=1}^{m} \mathbf{w}_t \|^2/(m+1).$$

The proof is completed by the standard mistake bound theorem and the following fact: $\max_{\mathbf{x}} \|\phi_t(\mathbf{x})\|^2 = 2 \max_{\mathbf{x}} \|\mathbf{x}\|^2 \leq 2R^2$. □

3.2 Mistake Bound Comparison

In the following, we would like to explore under what circumstances the **FE** framework can work better than individual models and the **S+T** framework using Theorem 1. The analysis is done based on the assumption that all frameworks use the perceptron algorithm.

Before showing the bound analysis, note that the framework proposed by (Evgeniou and Pontil, 2004; Finkel and Manning, 2009) is a generalization over these three frameworks (**FE**, **S+T**, and the baseline)[6]. However, our goal in this paper is different: we try to provide a deep discussion on *when and why* one should use a particular framework.

Here, we compare the mistake bounds of the feature sharing framework to that of the baseline approach, which learns each task independently[7]. In order to make the comparison easier, we make some simplifying assumptions. First, we assume that the problem contains only two tasks, 1 and 2. We also assume that $\|\mathbf{w}_1\| = \|\mathbf{w}_2\| = a$. These assumptions greatly reduce the complexity of the analysis and can give us greater insight into the comparisons.

Following the assumptions and Theorem 1, the mistake bound for the FE frameworks is

$$4(2 - \cos(\mathbf{w}_1, \mathbf{w}_2))R^2 a^2/(3\mu^2) \qquad (4)$$

This line of analysis leads to interesting bound comparisons for two cases. In the first case, we assume that task 1 and task 2 are essentially the same. In the second, more common case, we assume that they are different.

First, when we know a priori that task 1 and task 2 are essentially the same, we can combine the training data from the two tasks and train them as a single task. Therefore, given that we do not need to expand the feature space, the number of mistakes is now bounded by $R^2 a^2/\mu^2$. Note that this bound is in fact better than (4) with $\cos(\mathbf{w}_1, \mathbf{w}_2) = 1$. Therefore, if we know a priori that these two tasks are the same, training a single model is better than using the feature shared approach.

In practice, it is often the case that the two tasks are not the same. In this case, the number of mistakes of an independent approach on *both* task 1 and 2 will be bounded by the summation of the mistake bounds of task 1 and task 2. Therefore, using the independent approach, the number of mistakes for the perceptron algorithm on *both* tasks is bounded by $2R^2 a^2/\mu^2$. The following results can be obtained by directly comparing the two bounds,

Corollary 1. *Assume there exists* $\mathbf{w}_1$ *and* $\mathbf{w}_2$ *which separate* $\mathcal{D}_1$ *and* $\mathcal{D}_2$ *respectively with functional margin* μ*, and* $\|\mathbf{w}_1\| = \|\mathbf{w}_2\| = a$*. In this case: (4) will be smaller than the bound of individual approach,* $2R^2 a^2/\mu^2$*, if and only if* $\cos(\mathbf{w}_1, \mathbf{w}_2) = (\mathbf{w}_1^T \mathbf{w}_2)/(\|\mathbf{w}_1\|\|\mathbf{w}_2\|) > \frac{1}{2}$.

If we assume that there is no difference in $P(X)$ between domains and hence we can treat $\cos(\mathbf{w}_1, \mathbf{w}_2)$ as the similarity between two tasks, the above argument suggests:

- If the two tasks are very different, the baseline approach (building two models) is better than **FE** and **S+T**.

[6] The framework proposed by (Evgeniou and Pontil, 2004; Finkel and Manning, 2009) is a generalization of Eq. (1). It allows the user to weight each block of features. If we put zero weight on the shared block, it becomes the baseline approach. On the other hand, if we put zero weight on all task-specific blocks, the framework becomes the **S+T** approach.

[7] Note that mistake bound results can be generalized to generalization bound results. See (Zhang, 2002).

- If the tasks are similar enough, **FE** is better than baseline and **S+T**.
- If the tasks are almost the same, **S+T** becomes better than **FE** and baseline.

In Section 4.1, we will evaluate whether these claims can be justified empirically.

4 Artificial Data Experiment Study

In this section we will present artificial experiments. We have two primary goals: 1) verifying the analysis proposed in Section 3, and 2) showing that the representation shift will impact the behavior of the **FE** algorithm. The second point will be verified again in the real world experiments in Section 5.

Data Generation In the following artificial experiments we experiment with domain adaptation by generating training and test data for two tasks, source and target, where we can control the difference between task definitions. The general procedure can be divided into two steps: 1) generating weight vectors $\mathbf{z}_1$ and $\mathbf{z}_2$ (for source and target respectively), and 2) randomly generating labeled instances for training and testing using $\mathbf{z}_1$ and $\mathbf{z}_2$.

The different experiments start with the same basic $\mathbf{z}_1$ and $\mathbf{z}_2$, but then may alter these weights to introduce task dissimilarities or similarities. The basic $\mathbf{z}_1$ and $\mathbf{z}_2$ are both generated by a multivariate Gaussian distribution with mean $\mathbf{z}$ and a diagonal covariance matrix βI:

$$\mathbf{z}_1 \sim \mathcal{N}(\mathbf{z}, \beta I), \mathbf{z}_2 \sim \mathcal{N}(\mathbf{z}, \beta I),$$

where $\mathcal{N}$ is the normal distribution and $\mathbf{z}$ is random vector with zero mean. Note that $\mathbf{z}$ is only used to generate $\mathbf{z}_1$ and $\mathbf{z}_2$. There is one parameter, β, that controls the variance of the Gaussian distribution. Hence we use β to roughly control the "angle" of $\mathbf{z}_1$ and $\mathbf{z}_2$. When β is close to zero, $\mathbf{z}_1$ and $\mathbf{z}_2$ will be very similar. On the other hand, when β is large, $\mathbf{z}_1$ and $\mathbf{z}_2$ can be very different. In these experiments, we vary β between 0.01 and 5 so that we are experimenting only with tasks where the weight the task difference is the "angle" or cosine between $\mathbf{z}_1$ and $\mathbf{z}_2$. Once we obtain the $\mathbf{z}_1$ and $\mathbf{z}_2$, we normalize them to the unit length.

After selecting $\mathbf{z}_1$ and $\mathbf{z}_2$, we then generate labeled instances $(\mathbf{x}, y)$ for the source task in the following way. For each example $\mathbf{x}$, we randomly generate n binary features, where each feature has 20% chance to be active. We then label the example by

$$y = sign(\mathbf{z}_1^T \mathbf{x}),$$

The data for the target task is generated similarly with $\mathbf{z}_2$. In these experiments, we fix the number of features n to be 500 and generate 100 source training examples and 40 target training examples, along with 1000 target testing examples. This matches the reasonable case in NLP where there are more features than training examples and each feature vector is sparse. In all of the experiments, we report the averaged testing error rate on the target testing data.

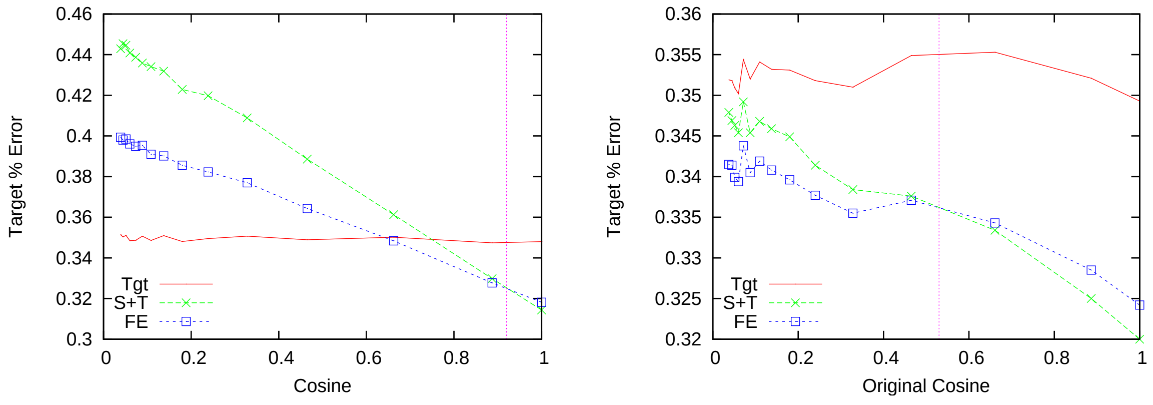
4.1 Experiment 1, FE algorithm

Goal The goal here is to verify our theoretical analysis in Section 3. Note that we do not introduce representation shift in this experiment and assume that both source and target domains use exactly the same features.

Result Figure 1(a) shows the performance of the three training algorithms as variance decreases and thus cosine between weight vectors (or measure of task similarity) goes to 1. Note that **FE** labeled adaptation framework beats **TGT** once the task cosine passes approximately 0.6. Initially FE slightly outperforms **S+T** until the tasks are close enough together that it is better to treat all the data as coming from one task. Note that while the experiments are based on the adaptation setting, the results match our analysis based on the multitask setting in Section 3.

4.2 Experiment 2, Unseen Features

Goal So far we have not considered the difference in $P(X)$ between domains. In the previous experiment, we used only cosine as our task similarity measurement to decide what is the best framework. However, task similarity should consider the difference in both $P(X)$ and $P(Y|X)$, and the cosine measurement is not sufficient for this. Here we construct a simple example to show that even a simple representation shift can change the behavior of the labeled adaptation framework. This case shows that **S+T** can be better than **FE** even when the tasks are not similar according to the cosine measurement.



(a) Basic Similarity

(b) Shared Features

Figure 1: Artificial Experiment comparing labeled adaptation performance vs. cosine between base weight vectors that defines two tasks, before and after cross-domain shared features are added. Figure (a) shows results from experiment 1. For **FE** adaptation algorithm to work the tasks need to be close (cosine $>$ 0.6), and if the tasks are close enough (cosine $\approx$ 1, dividing line) then it is better to just pool source and target training data together (the **S+T** algorithm). Figure (b) shows results for experiment 3 when shared features are added to the base weight vectors as used in experiment 1. Here the cosine similarity measure is between the base task weight vectors before the shared features have been added. Both labeled adaptation algorithms effectively use the shared features to improve over just training on target. With shared features added the dividing line where **S+T** improves over **FE** decreases so even for tasks that are initially further apart, once clusters are added the **S+T** algorithm does better than **FE**. Each point represents the average of 2000 training runs with random initial $\mathbf{z}_1$ and $\mathbf{z}_2$ generating data.

Result The second experiment deals with the case where features may appear in only one domain but should be treated like known features in the other domain. An example of this are out of vocabulary words that may not exist in a small target training task, but have synonyms in the source training data. In this case if we had features grouping words (say by word meanings) then we would recover this cross-domain information. In this experiment we want to explore which adaptation algorithm performs best *before* these features are applied.

To simulate this case we start with similar weight vectors $\mathbf{z}_1$ and $\mathbf{z}_2$ (sampled with variance = 0.00001, cos($\mathbf{z}_1$,$\mathbf{z}_2$) $\approx$ 1), but then shift some set of dimensions so that they represent features that appear only in one domain.

$$\mathbf{z}_1 = (\mathbf{a}_1, \mathbf{b}_1) \rightarrow \mathbf{z}'_1 = (\mathbf{0}, \mathbf{b}_1, \mathbf{a}_1)$$
$$\mathbf{z}_2 = (\mathbf{a}_2, \mathbf{b}_2) \rightarrow \mathbf{z}'_2 = (\mathbf{a}_2, \mathbf{b}_2, \mathbf{0})$$

By changing the ratio of the size of the dissimilar subset $\mathbf{a}$ to the similar subset $\mathbf{b}$ we can make the two weight vectors $\mathbf{z}'_1$ and $\mathbf{z}'_2$ more or less similar. Using these two new weight vectors we can proceed as above, generating training and testing data.

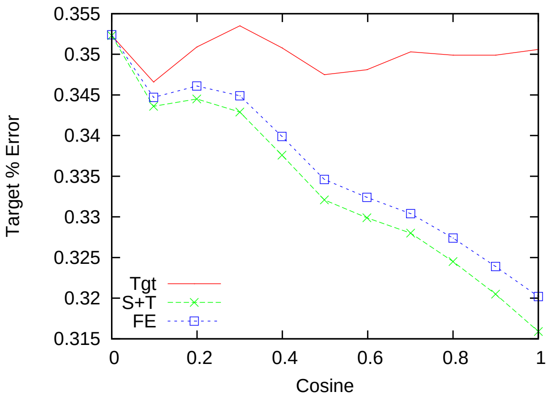


Figure 2: Artificial Experiment where unknown features are included in source or target domains, but not the other. The simple **S+T** adaptation framework is best able to exploit the set of shared features so performs best over the whole space of similarity in this setting.

Figure 2 shows the performance of the three algorithms on this data as the number of unrelated features are decreased. Over the entire range the combined algorithm **S+T** does better since it more efficiently exploits the shared similar $\mathbf{b}$ subset of the feature space. When the **FE** algorithm tries to create the shared features, it considers both the similar subset $\mathbf{b}$ and dissimilar subset $\mathbf{a}$. However, since $\mathbf{a}$ should not be shared, **FE** algorithm becomes less effective than the **S+T** algorithm. See the bound comparison in Section 3.2 for more intuitions. With this experiment we have demonstrated that there is a need to consider label and unlabeled adaptation frameworks together.

4.3 Experiment 3, Shared Features

Goal A good unlabeled adaptation framework should try to find features that "work" across domains. However, it is not clear how these newly added features will impact the behavior of the labeled adaptation frameworks. In this experiment, we show that the new shared features will bring the domains together, and hence make **S+T** a very strong adaptation framework.

Result For the third experiment we start with the same setup as in the first experiment, but then augment the initial weight vector with additional shared weights. These shared weights correspond to the introduction of features that appear in both domains and have the same meaning relative to the tasks, the ideal result of unlabeled adaptation methods.

To generate this case we again start with $\mathbf{z}_1$ and $\mathbf{z}_2$ of varying similarity as in section 4.1, then generate a random weight vector for shared features and append this to both weight vectors.

$$\mathbf{z}_s \sim \mathcal{N}(0, I), \mathbf{z}_1'' = (\mathbf{z}_1, \gamma\mathbf{z}_s), \mathbf{z}_2'' = (\mathbf{z}_2, \gamma\mathbf{z}_s),$$

where γ is used to put increased importance on the shared weight vectors by increasing the total weight of that section relative to the base $\mathbf{z}_1$ and $\mathbf{z}_2$ subsets. In our experiments we use 100 shared features to the 500 base features and set γ to 2.

Figure 1(b) shows the performance of the labeled adaptation algorithms once shared features had been added. Here the x-axis is the cosine between the original task weight vectors, demonstrating how the shared features improve performance on potentially dissimilar tasks. Whereas in the first experiment **FE** does not improve over just training on target data until the cosine is greater than 0.6, once shared features have been added then both **FE** and **S+T** use these features to learn with originally dissimilar tasks. Furthermore the shared features tend to push the tasks 'closer' so that **S+T** improves over **FE** earlier. Comparing to Figure 1(a), there are regions where before shared features are added it is better to use **FE**, and after shared features are added it is better to use **S+T**. This shows that labeled adaptation and unlabeled are *not* independent. Therefore, it is important to combine these two aspects to see the real contribution of each adaptation framework.

In these three artificial experiments we have demonstrated cases where both **FE** or **S+T** are the best algorithm before and after representation changes like those created with unlabeled adaptation are imposed. This fact points to the perhaps obvious conclusion that there is not a single best adaptation algorithm, and the determination of specific best practices depends on task similarity (in both $P(X)$ and $P(Y|X)$), especially after being brought closer together with other adaptation approaches. If there is one common trend it is that often once two tasks have been brought close together using a shared representation, then the tasks are now close enough such that the simple **S+T** algorithm does well.

5 Real World Experiments

In Section 4, we have shown through artificial data experiments that labeled and unlabeled adaptation algorithms are not independent. In this section, we focus on experiments with real datasets.

For the labeled adaptation algorithms, we have the following options:

- **TGT**: Only uses target labeled training dataset.
- **FE**: Uses both labeled datasets.
- **FE$^+$**: Uses both labeled datasets. A modification of the FE algorithm, equivalent to multiplying the "shared" part of the FE feature vector (Eq. (1)) by 10 (Finkel and Manning, 2009).
- **S+T**: Uses both source and target labeled datasets to train a single model with all labeled data directly.

Throughout all of our experiments, we use SVMs trained with a modified java implementation[8] of LIBLINEAR as our underlying learning classifier (Hsieh et al., 2008). For the tasks that require structures, we model each individual decision using

[8] Our code is modified from the version available on `http://www.bwaldvogel.de/liblinear-java/`

Algorithm	**TGT**	**FE**	**FE$^+$**	**S+T**
SRC labeled data?	no	yes		
Target labeled data	Token F1			
(a) MUC7 Dev	58.6	70.5	**74.3**	73.1
(a) + cluster	77.5	82.5	**83.3**	**83.3**
(b) MUC7 Train	73.0	78.2	**80.1**	78.7
(b) + cluster	85.4	86.4	86.2	**86.5**

Table 2: **NER Experiments.** We bold face the best accuracy in a row and underline the runner up. Both unlabeled adaptation algorithms (adding cluster features) and labeled adaptation algorithm (using source labeled data) help the performance significantly. Moreover, adding cluster-like features also changes the behavior of the labeled adaptation algorithms. Note that after adding cluster features, **S+T** becomes quite competitive with (or slightly better than) the **FE$^+$** approach. The size of MUC7 develop set is roughly 20% of the size of the MUC7 training set.

a local SVM classifier then make our prediction using a greedy approach from left to right. While we could use a more complex model such as Conditional Random Field (Lafferty et al., 2001), as we will see later, our simple model generates state-of-the-art results for many tasks. Regarding parameter selection, we selected the SVM regularization parameter for the baseline model (**TGT**) and then fix it for all algorithms[9].

Named Entity Recognition Our first task is Named Entity Recognition (NER). The source domain is from the CoNLL03 shared task (Tjong Kim Sang and De Meulder, 2003) and the target domain is from the MUC7 dataset. The goal of this adaptation system is to maximize the performance on the test data of MUC7 dataset with CoNLL training data and (some) MUC7 labeled data. As an unlabeled adaptation method to address feature sparsity, we add cluster-like features based on the gazetteers and word clustering resources used in (Ratinov and Roth, 2009) to bridge the source and target domain. We experiment with both MUC development and training set as our target labeled sets.

The experimental results are in Table 2. First, notice that addressing the feature sparsity issue helps the performance significantly. Adding cluster-like features improves the Token-F1 by around 10%. On the other hand, adding target labeled data also helps the results significantly. Moreover, using both target labeled data and cluster-like shared representation are mutually beneficial in all cases.

Importantly, adding cluster-like features changes the behavior of the labeled adaptation algorithms. When the cluster-like features are not added, the **FE$^+$** algorithm is in general the best labeled adaptation framework. This result agrees with the results showed in (Finkel and Manning, 2009), where the authors show that **FE$^+$** is the best labeled adaptation framework in their settings. However, after adding the cluster-like features, the simple **S+T** approach becomes very competitive to both **FE** and **FE$^+$**. This matches our analysis in Section 4: *resolving features sparsity will change the behavior of labeled adaptation frameworks.*

We compare the simple **S+T** algorithm with cluster-like features to other published results on adapting from CoNLL dataset to MUC7 dataset in table 3. Past works on this setting often only focus on one class of adaption approach. For example, (Ratinov and Roth, 2009) only use the cluster-like features to address the feature sparsity problem, and (Finkel and Manning, 2009) only use target labeled data without using gazetteers and word-cluster information. Notice that because of combining two classes of adaption algorithms, our approach is significantly better than these two systems[10].

Preposition Sense Disambiguation We also test the combination of unlabeled and labeled adaption on the task of Preposition Sense Disambiguation. Here the data contains multiple prepositions where each preposition has many different senses. The goal is to predict the right sense for a given preposition in the testing data. The source domain is the SemEval 2007 preposition WSD Task and the target domain is from the dataset annotated in (Dahlmeier et al., 2009). Our feature design mainly comes from (Tratz and Hovy, 2009) (who do not evaluate their system on our target data). As our un-

[9] We use L2-hinge loss for all of the experiments, with $C = 2^{-4}$ for NER experiments and $C = 2^{-5}$ for the PSD experiments.

[10] The work (Ratinov and Roth, 2009) also combines their system with several document-level features. While it is possible to add these features in our system, we do not include any global features for the sake of simplicity. Note that our system is competitive to (Ratinov and Roth, 2009) even though our system does not use global features.

Systems	Cluster?	TGT?	P.F1	T.F1
Our NER	y	y	**84.1**	**86.5**
FM09	n	y	79.98	N/A
RR09	y	n	N/A	83.2
RR09 + global	y	n	N/A	86.2

Table 3: Comparisons between different NER systems. P.F1 and T.F1 represent the phrase-level and token-level F1 score, respectively. We use "Cluster?" to indicate if cluster features are used and use "TGT?" to indicate if target labeled data is used. Previous systems often only use one class of adaptation algorithms. Using both adaptation aspects makes our system perform significantly better than FM09 and RR09.

Algorithm	**TGT**	**FE**	**FE**$^+$	**S+T**
SRC labeled data?	no	yes		
Target labeled data	Accuracy			
10% Tgt	43.8	48.2	**51.3**	49.7
10% Tgt + Cluster	44.9	50.5	51.8	**52.0**
100% Tgt	59.5	60.5	60.3	**61.2**
100% Tgt + Cluster	61.3	62.0	61.2	**62.1**

Table 4: **Preposition Sense Disambiguation**. We mark the best accuracy in a row using the bold font and underline the runner up. Note that both adding cluster features and adding source labeled data help the performance significantly. Moreover, adding clusters also changes the behavior of the labeled adaptation algorithms.

labeled adaptation approach we augment all word based features with cluster information from separately generated hierarchical Brown clusters (Brown et al., 1992).

The experimental results are in Table 4. Note that we see phenomena similar to what happened in the NER experiments. First, both labeled and unlabeled adaptation improves the system. When only 10% of the target labeled data is used, the inclusion of the source labeled data helps significantly. When there is more labeled data, labeled and unlabeled adaption have similar impact. Again, using unlabeled adaption changes the behavior of the labeled adaption algorithms.

In Table 5, we compare our system to (Dahlmeier et al., 2009), who do not use the SemEval data but jointly train their preposition sense disambiguation system with a semantic role labeling system. With both labeled and unlabeled adaption, our system is significantly better.

Systems	ACC
Our PSD (**S**+**T** and cluster)	62.1
DNS09	56.5
DNS09 + SRL	58.8

Table 5: Comparison between different PSD systems. Note that after adding cluster features and source labeled data with **S+T** approach, our system outperforms the state-of-the-art system proposed in (Dahlmeier et al., 2009), even though they jointly learn a PSD and SRL system together.

6 Conclusion

In this paper, we point out the necessities of combining labeled and unlabeled adaptation algorithms. We analyzed the FE algorithm both theoretically and empirically, demonstrating that it requires both a minimal amount of task similarity to work, and past a certain level of similarity other, simpler approaches are better. More importantly, through artificial data experiments we found that applying unlabeled adaptation algorithms may change the behavior of labeled adaptation algorithms as representations change, and hence affect the choice of labeled adaptation algorithm. Experiments with real-world datasets confirmed that combinations of both adaptation methods provide the best results, often allowing the use of simple labeled adaptation approaches. In the future, we hope to develop a joint algorithm which addresses both labeled and unlabeled adaptation at the same time.

Acknowledgment We thank Vivek Srikumar for providing the baseline implementation of preposition sense disambiguation. We also thank anonymous reviewers for their useful comments. University of Illinois gratefully acknowledges the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *J. Mach. Learn. Res.*

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*.

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*.

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*.

D. Dahlmeier, H. T. Ng, and T. Schultz. 2009. Joint learning of preposition senses and semantic roles of prepositional phrases. In *EMNLP*.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *ACL*.

T. Evgeniou and M. Pontil. 2004. Regularized multi–task learning. In *KDD*.

J. R. Finkel and C. D. Manning. 2009. Hierarchical bayesian domain adaptation. In *NAACL*.

C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. 2008. A dual coordinate descent method for large-scale linear svm. In *ICML*.

Fei Huang and Alexander Yates. 2009. Distributional representations for handling sparsity in supervised sequence-labeling. In *ACL*.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *ACL*.

T. Koo, X. Carreras, and M. Collins. 2008. Simple semi-supervised dependency parsing. In *ACL*.

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

P. Liang. 2005. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology.

A. Novikoff. 1963. On convergence proofs for perceptrons. In *Proceeding of the Symposium on the Mathematical Theory of Automata*.

L. Ratinov and D. Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*.

S. Tratz and D. Hovy. 2009. Disambiguation of preposition sense using linguistically motivated features. In *NAACL*.

Tong Zhang. 2002. Covering number bounds of certain regularized linear function classes. *J. Mach. Learn. Res.*

Training continuous space language models: some practical issues

Le Hai Son and **Alexandre Allauzen** and **Guillaume Wisniewski** and **François Yvon**
Univ. Paris-Sud, France and LIMSI/CNRS
BP 133, 91403 Orsay Cedex
Firstname.Lastname@limsi.fr

Abstract

Using multi-layer neural networks to estimate the probabilities of word sequences is a promising research area in statistical language modeling, with applications in speech recognition and statistical machine translation. However, training such models for large vocabulary tasks is computationally challenging which does not scale easily to the huge corpora that are nowadays available. In this work, we study the performance and behavior of two neural statistical language models so as to highlight some important caveats of the classical training algorithms. The induced word embeddings for extreme cases are also analysed, thus providing insight into the convergence issues. A new initialization scheme and new training techniques are then introduced. These methods are shown to greatly reduce the training time and to significantly improve performance, both in terms of perplexity and on a large-scale translation task.

1 Introduction

Statistical language models play an important role in many practical applications, such as machine translation and automatic speech recognition. Let $\mathcal{V}$ be a finite vocabulary, statistical language models define distributions over sequences of words w_1^L in $\mathcal{V}^\star$ usually factorized as:

$$P(w_1^L) = P(w_1) \prod_{l=1}^{L} P(w_l | w_1^{l-1})$$

Modeling the joint distribution of several discrete random variables (such as words in a sentence) is difficult, especially in real-world Natural Language Processing applications where $\mathcal{V}$ typically contains dozens of thousands words.

Many approaches to this problem have been proposed over the last decades, the most widely used being back-off n-gram language models. n-gram models rely on a Markovian assumption, and despite this simplification, the maximum likelihood estimate (MLE) remains unreliable and tends to underestimate the probability of very rare n-grams, which are hardly observed even in huge corpora. Conventional smoothing techniques, such as Kneser-Ney and Witten-Bell back-off schemes (see (Chen and Goodman, 1996) for an empirical overview, and (Teh, 2006) for a Bayesian interpretation), perform back-off on lower order distributions to provide an estimate for the probability of these unseen events. n-gram language models rely on a discrete space representation of the vocabulary, where each word is associated with a discrete index. In this model, the morphological, syntactic and semantic relationships which structure the lexicon are completely ignored, which negatively impact the generalization performance of the model. Various approaches have proposed to overcome this limitation, notably the use of word-classes (Brown et al., 1992; Niesler, 1997), of generalized back-off strategies (Bilmes et al., 1997) or the explicit integration of morphological information in the random-forest model (Xu and Jelinek, 2004; Oparin et al., 2008).

One of the most successful alternative to date is to use *distributed word representations* (Bengio et al., 2003), where distributionally similar words are represented as neighbors in a continuous space. This

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 778–788,
MIT, Massachusetts, USA, 9-11 October 2010.

turns n-grams distributions into smooth functions of the word representations. These representations and the associated probability estimates are jointly computed in a multi-layer neural network architecture. This approach has showed significant and consistent improvements when applied to automatic speech recognition (Schwenk, 2007; Emami and Mangu, 2007; Kuo et al., 2010) and machine translation tasks (Schwenk et al., 2006). Hence, continuous space language models are becoming increasingly used. These successes have revitalized the research on neuronal architectures for language models, and given rise to several new proposals (see, for instance, (Mnih and Hinton, 2007; Mnih and Hinton, 2008; Collobert and Weston, 2008)). A major difficulty with these approaches remains the complexity of training, which does not scale well to the massive corpora that are nowadays available. Practical solutions to this problem are discussed in (Schwenk, 2007), which introduces a number of optimization and tricks to make training doable. Even then, training a neuronal language model typically takes days.

In this paper, we empirically study the convergence behavior of two multi-layer neural networks for statistical language modeling, comparing the standard model of (Bengio et al., 2003) with the log-bilinear (LBL) model of (Mnih and Hinton, 2007). Our contributions are the following: we first propose a reformulation of Mnih and Hinton's model, which reveals its similarity with extant models, and allows a direct and fair comparison with the standard model. For the standard model, these results highlight the impact of parameter initialization. We first investigate a re-initialization method which allows to escape from the local extremum the standard model converges to. While this method yields a significative improvement, the underlying assumption about the structure of the model does not meet the requirement of very large-scale tasks. We therefore introduce a different initialization strategy, called *one vector initialization*. Experimental results show that these novel training strategies drastically reduce the total training time, while delivering significant improvements both in terms of perplexity and in a large-scale translation task.

The rest of this paper is organized as follows. We first describe, in Section 2, the standard and the LBL language models. By reformulating the latter, we show that both models are very similar and emphasize the remaining differences. Section 2.4 discusses complexity issues and possible solutions to reduce the training time. We then report, in Section 3, preliminary experimental results that enlighten some caveats of the standard approach. Based on these observations, we introduce in Section 4 novel and more efficient training schemes, yielding improved performance and a reduced training time both on small and large scale experiments.

2 Continuous space language models

Learning a language model amounts to estimate the parameters of the discrete conditional distribution over words given each possible history, where the history corresponds to some function of the preceding words. For an n-gram model, the history contains the $n-1$ preceding words, and the model parameters correspond to $P(w_l|w_{l-n+1}^{l-1})$. Continuous space language models aim at computing these estimates based on a distributed representation of words (Bengio et al., 2003), thereby reducing the sparsity issues that plague conventional maximum likelihood estimation. In this approach, each word in the vocabulary is mapped into a real-valued vector and the conditional probability distributions are then expressed as a (parameterized) smooth function of these feature vectors. The formalism of neural networks allows to express these two steps in a well-known framework, where, crucially, the mapping and the model parameters can be learned in conjunction. In the next paragraphs, we describe the two continuous space language models considered in our study and present the various issues associated with the training of such models, as well as their most common remedies.

2.1 The standard model

In the following, we will consider words as indices in a finite dictionary of size V; depending on the context, w will either refer to the word or to its index in the dictionary. A word w can also be represented by a 1-of-V coding vector $\mathbf{v}$ of $\mathbb{R}^V$ in which all elements are null except the w^{th}. In the standard approach of (Bengio et al., 2003), the feed-forward network takes as input the $n-1$ word history and delivers an estimate of the probability $P(w_l|w_{l-n+1}^{l-1})$

as its output. It consists of three layers.

The first layer builds a continuous representation of the history by mapping each word into its real-valued representation. This mapping is defined by $\mathbf{R}^T\mathbf{v}$, where $\mathbf{R} \in \mathbb{R}^{V \times m}$ is a *projection matrix* and m is the dimension of the continuous projection word space. The output of this layer is a vector $\mathbf{i}$ of $(n-1)m$ real numbers obtained by concatenating the representations of the context words. The projection matrix $\mathbf{R}$ is shared along all positions in the history vector and is learned automatically.

The second layer introduces a non-linear transform, where the output layer activation values are defined by $\mathbf{h} = \tanh\left(\mathbf{W_{ih}}\mathbf{i} + \mathbf{b_{ih}}\right)$, where $\mathbf{i}$ is the input vector, $\mathbf{W_{ih}} \in \mathbb{R}^{H \times (n-1)m}$ and $\mathbf{b_{ih}} \in \mathbb{R}^H$ are the parameters of this layer. The vector $\mathbf{h} \in \mathbb{R}^H$ can be considered as an higher (more abstract) representation of the context than $\mathbf{i}$.

The third layer is an output layer that estimates the desired probability, thanks to the *softmax* function:

$$\begin{aligned} P(w_l = k | w_{l-n+1}^{l-1}) &= \frac{\exp(\mathbf{o_k})}{\sum_{k'} \exp(\mathbf{o_{k'}})} \quad (1) \\ \mathbf{o} &= \mathbf{W_{ho}}\mathbf{h} + \mathbf{b_{ho}}, \quad (2) \end{aligned}$$

where $\mathbf{W_{ho}} \in \mathbb{R}^{V \times H}$ and $\mathbf{b_{ho}} \in \mathbb{R}^V$ are respectively the projection matrix and the bias term associated with this layer. The w^{th} component in $\mathbf{P}$ corresponds to the estimated probability of the w^{th} word of the vocabulary given the input history vector.

The standard model has two hyper-parameters (the dimension of projection space m and the size of hidden layer, H) that define the architecture of the neural network and a set of free parameters $\mathbf{\Theta}$ that need to be learned from data: the projection matrix $\mathbf{R}$, the weight matrix $\mathbf{W_{ih}}$, the bias vector $\mathbf{b_{ih}}$, the weight matrix $\mathbf{W_{ho}}$ and the bias vector $\mathbf{b_{ho}}$.

In this model, the projection matrices $\mathbf{R}$ and $\mathbf{W_{ho}}$ play similar roles as they define maps between the vocabulary and the hidden representation. The fact that $\mathbf{R}$ assigns similar representations to history words w_1 and w_2 implies that these words can be exchanged with little impact on the resulting probability distribution. Likewise, the similarity of two lines in $\mathbf{W_{ho}}$ is an indication that the corresponding words tend to have a similar behavior, i.e. tend to have a similar probabilities of occurrence in all contexts. In the remainder, we will therefore refer to $\mathbf{R}$ as the matrix representing the **context space**, and to $\mathbf{W_{ho}}$ as the matrix for the **prediction space**.

2.2 The log-bilinear model

The work reported (Mnih and Hinton, 2007) describes another parameterization of the architecture introduced in the previous section. This parameterization is based on Factored Restricted Boltzmann Machine. According to (Mnih and Hinton, 2007), this model, termed the *log-bilinear* language model (LBL), achieves, for large vocabulary tasks, better results in terms of perplexity than the standard model, even if the reasons beyond this improvement remain unclear.

In this section, we will describe this model and show how it relates to the standard model. The LBL model estimates the n-gram parameters by:

$$P(w_l | w_{l-n+1}^{l-1}) = \frac{\exp(-E(w_l; w_{l-n+1}^{l-1}))}{\sum_w \exp(-E(w; w_{l-n+1}^{l-1}))} \quad (3)$$

In this equation, E is an *energy function* defined as:

$$\begin{aligned} E(w_l; w_1^{l-1}) = &- \left(\sum_{k=l-n+1}^{l-1} \mathbf{v_k}^T \mathbf{R} \mathbf{C}_k^T \right) \mathbf{R}^T \mathbf{v}_l \\ & \qquad\qquad\qquad\qquad\qquad\qquad (4) \\ &- \mathbf{b_r}^T \mathbf{R}^T \mathbf{v}_l - \mathbf{b_v}^T \mathbf{v}_l \\ = &- \mathbf{v}_l^T \mathbf{R} \left(\sum_{k=l-n+1}^{l-1} \mathbf{C}_k \mathbf{R}^T \mathbf{v_k} + \mathbf{b_r} \right) \\ &- \mathbf{v}_l^T \mathbf{b_v} \qquad\qquad\qquad\qquad (5) \end{aligned}$$

where $\mathbf{R}$ is the projection matrix introduced above, $(\mathbf{v}_k)_{l-n+1 \leq k \leq l-1}$ are the 1-of-V coding vectors for the history words and $\mathbf{v}_l$ is the coding vector for w_l; $\mathbf{C}_k \in \mathbb{R}^{m \times m}$ is a combination matrix and $\mathbf{b_r}$ and $\mathbf{b_v}$ denote bias vectors. All these parameters need to be learned during training.

Equation (4) can be rewritten using the notations introduced for the standard model. We then rename $\mathbf{b_r}$ and $\mathbf{b_v}$ respectively $\mathbf{b_{ih}}$ and $\mathbf{b_{ho}}$. We also denote $\mathbf{i}$ the concatenation of the $(n-1)$ vectors $\mathbf{R}^T\mathbf{v_k}$; likewise $\mathbf{W_{ih}}$ denotes the $H \times (n-1)m$ matrix obtained by concatenating row-wise the $(n-1)$ matrices $\mathbf{C}_k$. With these new notations, equations (4) and (3) can be rewritten as:

$$\mathbf{h} = \mathbf{W_{ih}}\mathbf{i} + \mathbf{b_{ih}}$$
$$\mathbf{o} = \mathbf{R}\mathbf{h} + \mathbf{b_{ho}}$$
$$P(w_l = k|w_{l-n+1}^{l-1}) = \frac{\exp(\mathbf{o_k})}{\sum_{k'} \exp(\mathbf{o_{k'}})}$$

This formulation allows to highlight the similarity of the LBL model and the standard model. These two models differ only by the activation function of their hidden layer (linear for the LBL model and tangent hyperbolic for the standard model) and by their definition of the prediction space: for the LBL model, the context space and the prediction space are the same ($\mathbf{R} = \mathbf{W_{ho}}$, and thus $H = m$), while in the standard model, the prediction space is defined independently from the context space. This restriction drastically reduces the number of free parameters of the LBL model.

It is finally noteworthy to outline the similarity of this model with standard maximum entropy language models (Lau et al., 1993; Rosenfeld, 1996). Let $\mathbf{x}$ denote the binary vector formed by stacking the (n-1) 1-of-V encodings of the history words; then the conditional probability distributions estimated in the model are proportional to $\exp F(x)$, where F is an affine transform of $\mathbf{x}$. The main difference with MaxEnt language models are thus the restricted form of the feature functions, which only test one history word, and the particular representation of F, which is defined as:

$$F(\mathbf{x}) = \mathbf{R}\mathbf{W_{ih}}\mathbf{R'}^T\mathbf{v} + \mathbf{R}\mathbf{b_{ih}} + \mathbf{b_{ho}}$$

where, as before, $\mathbf{R}'$ is formed by concatenating $(n-1)$ copies of the projection matrix $\mathbf{R}$.

2.3 Training and inference

Training the two models introduced above can be achieved by maximizing the log-likelihood $\mathcal{L}$ of the parameters Θ. This optimization is usually performed by stochastic back-propagation as in (Bengio et al., 2003). For all our experiments, the learning rate is fixed at 5×10^{-3}. The learning weight decay and the the weight decay (respectively 1×10^{-9} and 0) seem to have a minor impact on the results. Learning starts with a random initialization of the parameters under the uniform distribution and converges to a local maximum of the log-likelihood function. Moreover, to prevent overfitting, an early stopping strategy is adopted: after each epoch, training is stopped when the likelihood of a validation set stops increasing.

2.4 Complexity issues

The main problem with neural language models is their computational complexity. For the two models presented in this section, the number of floating point operations needed to predict the label of a single example is[1]:

$$((n-1) \cdot m + 1) \times H + (H + 1) \times V \qquad (6)$$

where the first term of the sum corresponds to the computation of the hidden layer and the second one to the computation of the output layer. The projection of the context words amounts to select one row of the projection matrix $\mathbf{R}$, as the words are represented with a 1-of-V coding vector. We can therefore assume that the computation complexity of the first layer is negligible. Most of the computation time is thus spent in the output layer, which implies that the computing time grows linearly with the vocabulary size. Training these models for large scale tasks is therefore challenging, and a number of tricks have been introduced to make training and inference tractable (Schwenk and Gauvain, 2002; Schwenk, 2007).

Short list A simple method to reduce the complexity in inference and in learning is to reduce the size of the output vocabulary (Schwenk, 2007): rather than estimating the probability $P(w_l = w|w_{l-n+1}^{l-1})$ for all words in the vocabulary, we only estimate it for the N most frequent words of the training set (the so-called *short-list*). In this case, two vocabularies need to be considered, corresponding respectively to the *context vocabulary* $\mathcal{V}_c$ used to define the history; and the *prediction vocabulary* $\mathcal{V}_p$. However, this method fails to deliver any probability estimate for words outside of the prediction vocabulary, meaning that a fall-back strategy needs to be defined for those words. In practice, neural network

[1] Recall that learning requires to repeatedly predict the label for all the examples in the training set.

language models are combined with a conventional n-gram model as described in (Schwenk, 2007).

Batch mode and resampling Additional speed-ups can be obtained by propagating several examples at once through the network (Bilmes et al., 1997). This "batch mode" allows to factorize the matrix operations and cut down both inference and training time. In all our experiments, we used a batch size of 64. Moreover, the training time is linear in the number of examples in the training data[2]. Training on very large corpora, which, nowadays, comprise billions of word tokens, cannot be performed exhaustively and requires to adopt *resampling* strategies, whereby, at each epoch, the system is trained with only a small random subset of the training data. This approach enables to effectively estimate neural language models on very large corpora; it has also been observed empirically that sampling the training data can increase the generalization performance (Schwenk, 2007).

3 A head-to-head comparison

In this section, we analyze a first experimental study of the two neural network language models introduced in Section 2 in order to better understand the differences between these models especially in terms of the word representations they induce. Based on this study, we will propose, in the next section, improvements of both the speed and the prediction capacity of the models. In all our experiments, 4-gram language models are used.

3.1 Corpus

The data we use for training is a large monolingual corpus, containing all the English texts in the parallel data of the Arabic to English NIST 2009 constrained task[3]. It consists of 176 millions word tokens with $532,557$ different word types as the size of vocabulary. The perplexity is computed with respect to the 2006 NIST test data, which is used here as our development data.

3.2 Convergence study

In a first experiment, we trained the two models in the same setting: we choose to consider a small vocabulary comprising the $10,000$ most frequent words. The same vocabulary is used to constrain the words occurring in the history and the words to be predicted. The size of hidden layer is set to $m = H = 200$, the history contains the 3 preceding words, we use a batch size of 64, a resampling rate of 5% and no weight decay.

Figure 1 displays the perplexity convergence curve measured on the development data for the standard and the LBL models[4]. The convergence perplexities after the combination with the standard back-off model are also provided for all the models in table 2 (see section 4.3). We can observe that the LBL model converges faster than the standard model: the latter needs 13 epochs to reach the stopping criteria, while the former only needs 6 epochs. However, upon convergence, the standard model reaches a lower perplexity than the LBL model.

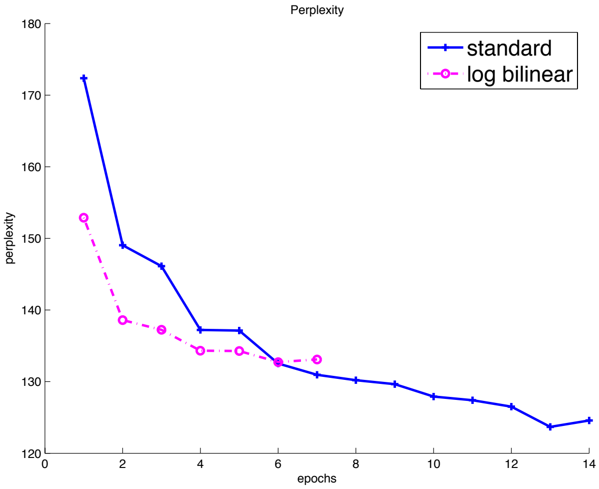


Figure 1: Convergence rate of the standard and the LBL models evaluated by the evolution of the perplexity on a development set

As described in Section 2.2, the main difference between the standard and the LBL model is the way the context and the prediction spaces are defined: in the standard model, the two spaces are distinct; in

[2] Equation (6) gives the complexity of inference for a single example.

[3] `http://www.itl.nist.gov/iad/mig/tests/mt/2009/MT09_ConstrainedResources.pdf`

[4] The use of a back-off 4-model estimated with the modified Knesser-Ney smoothing on the same training data achieves a perplexity of 141 on the development data.

the LBL model, they are bound to be the same. With a smaller number of parameters, the LBL model can not capture as many characteristics of the data as the standard model, but it converges faster[5]. This difference in convergence can be explained by the scarcity of the updates in the projection matrix $\mathbf{R}$ in the standard model: during backpropagation, only those weights that are associated with words in the history are updated. By contrast, each training sample updates all the weights in the prediction matrix $\mathbf{W_{ho}}$.

3.3 An analysis of the continuous word space

To deepen our understanding, we propose to further analyze the induced word embeddings by finding, for some randomly selected words, the five nearest neighbors (according to the Euclidian distance) in the context space and in the prediction space of the two models. Results are presented in Table 1.

If we look first at the standard model, the global picture is that for frequent words (*is*, *are*, and, to a lesser extend, *have*), both spaces seem to define meaningful neighborhood, corresponding to semantic and syntactic similarities; this is less true for rarer words, where we see a greater discrepancy between the context and prediction spaces. For instance, the date *1947* seems to be randomly associated in the context space, while the 5 nearest words in the prediction space form a consistent set of dates. The same trend is also observed for the word *Castro*. Our interpretation is that for less frequent words, the projection vectors are hardly ever updated and remain close to their original random initialization.

By contrast, the similarities in the (unique) projection space of the LBL remain consistent for all frequency ranges, and are very similar to the prediction space of the standard model. This seems to validate our hypothesis that in the standard model, the prediction space is learned much faster than the context space and corroborates our interpretation of the impact of the scarce updates of rare words. Another possible explanation is that there is no clear relation between the context space and the target function: the context space is learned only indirectly by backpropagation. As a result, due to the random initialization of the parameters and to data sparsity, many vectors of $\mathbf{R}$ might be blocked in some local maxima, meaning that similar vectors cannot be grouped in a consistent way and that the induced similarity is more “loose”.

4 Improving the standard model

In Section 3.2, we observed that slightly better results can be obtained with the standard rather than with the LBL model. The latter is however much faster to train, and seems to induce better projection matrices. Both effects can be attributed to the particular parameterization of this model, which uses the same projection matrix both for the context and for the prediction spaces. In this section, we propose several new learning regimes that allowed us to improve the standard model in terms of both speed and prediction capacity. All these improvements rely on the idea of sharing word representations. While this idea is not new (see for instance (Collobert and Weston, 2008)), our analysis enables to better understand its impact on the convergence rate. Finally, the improvements we propose are evaluated on a real-word machine translation task.

4.1 Improving performances with re-initialization

The experiments reported in the previous section suggest that it is possible to improve the performances of the standard model by building a better context space. Thus, we introduce a new learning regime, called *re-initialization* which aims to improve the context space by re-injecting the information on word neighborhoods that emerges in the prediction space. One possible implementation of this idea is as follows:

1. train a standard model until convergence;

2. use the prediction space of this model to initialize the context space of a new model; the prediction space is chosen randomly;

3. train this new model.

[5] We could increase the number of parameters of the LBL model for a fairer comparison with the standard model. However, this would also increase the size of the vocabulary and cause two new issues: on one hand, the time complexity would drastically increase for the LBL model, and on the other hand, both models would not be comparable in terms of perplexity as their vocabulary would be different.

Table 1: The 5 closest words in the representation spaces of the standard and LBL language models.

word (frequency)	*model*	*space*	*5 most closest words*
is	standard	context	was are were be been
900, 350	standard	prediction	was has would had will
	LBL	both	was reveals proves are ON
are	standard	context	were is was be been
478, 440	standard	prediction	were could will have can
	LBL	both	were is was FOR ON
have	standard	context	had has of also the
465, 417	standard	prediction	are were provide remain will
	LBL	both	had has Have were embrace
meeting	standard	context	meetings conference them 10 talks
150, 317	standard	prediction	undertaking seminar meetings gathering project
	LBL	both	meetings summit gathering festival hearing
Imam	standard	context	PCN rebellion 116. Cuba 49
787	standard	prediction	Castro Sen Nacional Al- Ross
	LBL	both	Salah Khaled Al- Muhammad Khalid
1947	standard	context	36 Mercosur definite 2002-2003 era
774	standard	prediction	1965 1945 1968 1964 1975
	LBL	both	1965 1976 1964 1968 1975
Castro	standard	context	exclusively 12. Boucher Zeng Kelly
768	standard	prediction	Singh Clark da Obasanjo Ross
	LBL	both	Clark Singh Sabri Rafsanjani Sen

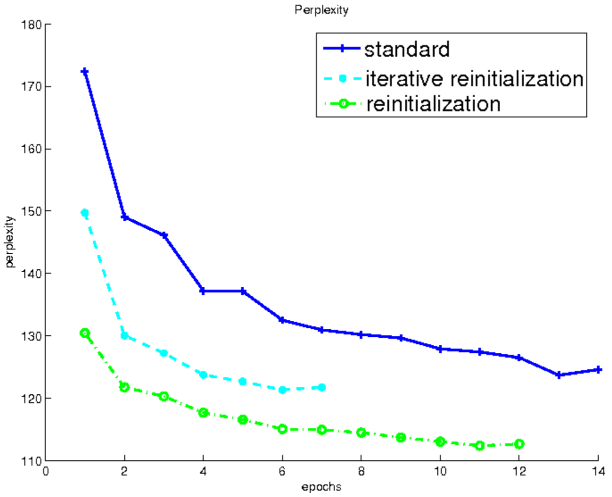


Figure 2: Evolution of the perplexity on a development set for various initialization regimes.

The evolution of the perplexity with respect to training epochs for this new method is plotted on Figure 2, where we only represent the evolution of the perplexity during the third training step. As can be seen, at convergence, the perplexity the model estimated with this technique is about 10% smaller than the perplexity of the standard model.

This result can be explained by considering the re-initialization as a form of annealing technique: re-initializing the context space allows to escape from the local extrema the standard model converges to. The fact that the prediction space provides a good initialization of the context space also confirms our analysis that one difficulty with the standard model is the estimation of the context space parameters.

4.2 Iterative re-initialization

The *re-initialization* policy introduced in the previous section significantly reduces the perplexity, at the expense of a longer training time, as it requires to successively train two models. As we now know that the parameters of the prediction space are faster to converge, we introduce a second training regime called *iterative re-initialization* which aims to take advantage of this property. We summarize this new training regime as follows:

1. Train the model for one epoch.
2. Use the prediction space parameters to reinitialize the context space.
3. Iterate steps (1) and (2) until convergence.

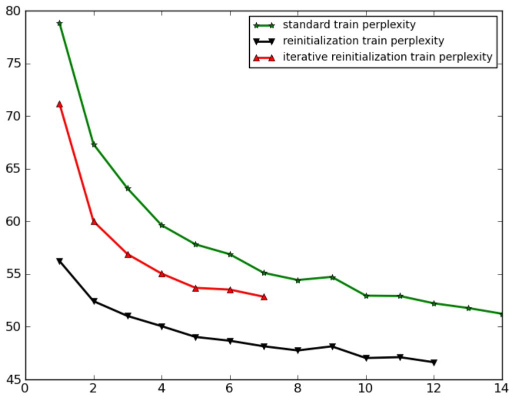


Figure 3: Evolution of the perplexity on the training data for various initialization regimes.

This regimes yields a model that is somewhat in-between the standard and LBL models as it adds a relationship between the two representation spaces, which lacks in the former model. This relationship is however not expressed through the tying of the corresponding parameters; instead we let the prediction space guide the convergence of the context space. As a consequence, we hope that it can achieve a convergence speed as fast as the one of the LBL model without degrading its prediction capacity.

The result plotted on Figure 2 shows that this indeed the case: using this training regime, we obtained a perplexity similar to the one of the standard model, while at the same time reducing the total training time by more than a half, which is of great practical interest (each epoch lasts approximately 8 hours on a 3GHz Xeon processor).

Figure 3 displays the perplexity convergence curve measured on the training data for the standard learning regime as well as for the *re-initialization* and *iterative re-initialization*. These results show the same trend as for the perplexity measured on the development data, and suggest a regularization effect of the re-initialization schemes rather than allowing the models to escape local optima.

4.3 One vector initialization

Principle The new training regimes introduced above outperform the standard training regime both in terms of perplexity and of training time. However, exchanging information between the context and prediction spaces is only possible when the same vocabulary is used in both spaces. As discussed in Section 2.4, this configuration is not realistic for very large-scale tasks. This is because increasing the number of predicted word types is much more computationally demanding than increasing the number of types in the context vocabulary. Thus, the former vocabulary is typically order of magnitudes larger than the latter, which means that the re-initialization strategies can no longer be directly used.

It is nonetheless possible to continue drawing inspirations from the observations made in Section 3, and, crucially, to question the random initialization strategy. As discussed above, this strategy may explain why the neighborhoods in the induced context space for the less frequent types were difficult to interpret. As a straightforward alternative, we consider a different initialization strategy where all the words in the context vocabulary are initially projected onto the same (random) point in the context space. The intuition is that it will be easier to build meaningful neighborhoods, especially for rare types, if all words are initially considered similar and only diverge if there is sufficient evidence in the training data to suggest that they should. This model is termed the *one vector initialization* model.

Experimental evaluation To validate this approach, we compare the convergence of a standard model trained (with the standard learning regime) with the one vector initialization regime. The context vocabulary is defined by the $532,557$ words occurring in the training data and the prediction vocabulary by the $10,000$ most frequent words[6]. All other parameters are the same as in the previous experiments. Based on the curves displayed on Figure 4, we can observe that the model obtained with the one vector initialization regime outperforms the model trained with a completely random initialization. Moreover, the latter reaches convergence in only 14 epochs, while the learning regime we propose only needs 9 epochs. Convergence is even faster than when we used the standard training regime and a small context vocabulary.

[6] In this case, the distinction between the *context* and the *prediction* vocabulary rules out the possibility of a relevant comparison based on perplexity between the continuous space language model and a standard back-off language model.

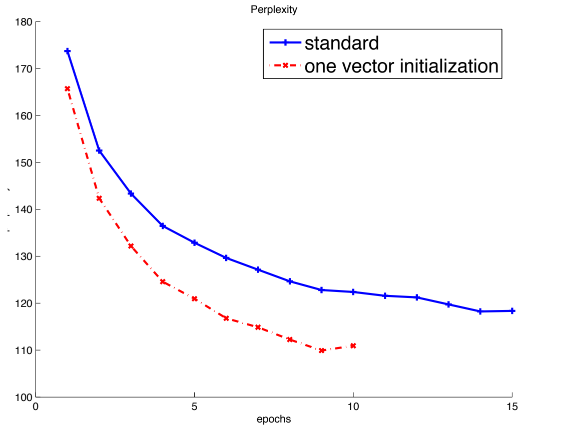
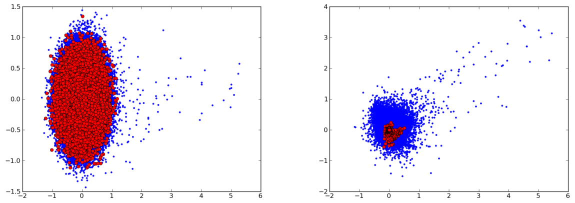


Figure 4: Perplexity with all-10, 000, 200 − 200 models

Table 2: Summary of the perplexity (*PPX*) results measured on the same development set with the different continuous space language models. For all of them, the probabilities are combined with the back-off n-gram model

$\mathcal{V}_c$ *size*	*Model*	*# epochs*	*PPX*
10000	log bilinear	6	239
	standard	13	227
	iterative reinit.	6	223
	reinit.	11	211
all	standard	14	276
	one vector init.	9	260



(a) with the standard model

(b) with the one vector initialization model

Figure 5: Comparison of the word embedding in the context space for numbers (red points).

To illustrate the impact of our initialization scheme, we also used a principal component analysis to represent the induced word representations in a two dimensional space. Figure 5 represents the vectors associated with numbers[7] in red, while all other words are represented in blue. Two different models are used: the standard model on the left, and the one vector initialization model on the right. We can observe that, for the standard model, most of the red points are scattered all over a large portion of the representation space. On the opposite, for the one vector initialization model, points associated with numbers are much more concentrated: this is simply because all the points are originally identical, and the training aim to spread the point around this starting point. We also created the closest word list reported in Table 3, in a manner similar to Table 1. Clearly, the new method seems to yield more meaningful neighborhoods in the context space.

It is finally noteworthy to mention that when used with a small context vocabulary (as in the experimental setting of Section 4.1) this initialization strategy underperforms the standard initialization. This is simply due to the much greater data sparsity in the large context vocabulary experiments, where the rarer word types are really rare (they typically occur once or twice). By contrast, the rarer words in the small vocabulary tasks occurred more than several hundreds times in the training corpus, which was more than sufficient to guide the model towards satisfactory projection matrices. This finally suggests that there still exists room for improvement if we can find more efficient initialization strategies than starting from one or several random points.

4.4 Statistical machine translation experiments

As a last experiment, we compare the various models on a large scale machine translation task. Statistical language models are key component of current statistical machine translation systems (Koehn, 2010), where they both help disambiguate lexical choices in the target language and influence the choice of the right word ordering. The integration of a neural network language model in such a system is far from easy, given the computational cost of computing word probabilities, a task that is performed repeatedly during the search of the best translation. We then had to resort to a two pass decoding approach: the first pass uses a conventional back-off language model to produce a n-best list (the n most likely translations and their associated scores); in the second pass, the probability of the neural language model is computed for each hypothesis and the n-

[7] Number are all the words consisting only of digits, with an optional sign, point or comma such as: *1947*; *0,001*; *-8,2*.

Table 3: The 5 closest words in the context space of the standard and one vector initialization language models

word (freq.)	*model*	*5 closest words*
is	standard	was are were been remains
$900,350$	1 vector init.	was are be were been
conducted	standard	undertaken launched $270,900 Mufamadi 6.44-km-long
$18,388$	1 vector init.	pursued conducts commissioned initiated executed
Cambodian	standard	Shyorongi $3,192,700 Zairian depreciations teachers
$2,381$	1 vector init.	Danish Latvian Estonian Belarussian Bangladeshi
automatically	standard	MSSD Sarvodaya $676,603,059 Kissana 2,652,627
$1,528$	1 vector init.	routinely occasionally invariably inadvertently seldom
Tosevski	standard	$12.3 Action,3 Kassouma 3536 Applique
34	1 vector init.	Shafei Garvalov Dostiev Bourloyannis-Vrailas Grandi
October-12	standard	39,572 anti-Hutu $12,852,200 non-contracting Party's
8	1 vector init.	March-26 April-11 October-1 June-30 August4
3727th	standard	Raqu Tatsei Ayatallah Mesyats Langlois
1	1 vector init.	4160th 3651st 3487th 3378th 3558th

best list is accordingly reordered to produce the final translations.

The different language models discussed in this article are evaluated on the Arabic to English NIST 2009 constrained task. For the continuous space language model, the training data consists in the parallel corpus used to train the translation model (previously described in section 3.1). The development data is again the 2006 NIST test set and the test data is the official 2008 NIST test set. Our system is built using the open-source Moses toolkit (Koehn et al., 2007) with default settings. To set up our baseline results, we used an extensively optimized standard back-off 4-grams language model using Kneser-Ney smoothing described in (Allauzen et al., 2009). The weights used during the reranking are tuned using the Minimum Error Rate Training algorithm (Och, 2003). Performance is measured based on the BLEU (Papineni et al., 2002) scores, which are reported in Table 4.

Table 4: BLEU scores on the NIST MT08 test set with different language models.

$\mathcal{V}_c$ size	*Model*	*# epochs*	*BLEU*
all	baseline	-	37.8
10000	log bilinear	6	38.2
	standard	13	38.3
	iterative reinit.	6	38.4
	reinit.	11	38.4
all	standard	14	38.6
	one vector init.	9	**38.7**

All the experimented neural language models yield to a significant BLEU increase. The best result is obtained by the one vector initialization standard model which achieves a 0.9 BLEU improvement. While this results is similar to the one obtained with the standard model, the training time is reduced here by a third.

5 Conclusion

In this work, we proposed three new methods for training neural network language models and showed their efficiency both in terms of computational complexity and generalization performance in a real-word machine translation task. These methods rely on conclusions drawn from a careful study of the convergence rate of two state-of-the-art models and are based on the idea of sharing the distributed word representations during training.

Our work highlights the impact of the initialization and the training scheme for neural network language models. Both our experimental results and our new training methods can be closely related to the pre-training techniques introduced by (Hinton and Salakhutdinov, 2006). Our future work will thus aim at studying the connections between our empirical observations and the deep learning framework.

Acknowledgments

This work was partly realized as part of the Quaero Program, funded by OSEO, the French agency for innovation.

References

Alexandre Allauzen, Josep Crego, Aurélien Max, and François Yvon. 2009. LIMSI's statistical translation systems for WMT'09. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 100–104, Athens, Greece, March. Association for Computational Linguistics.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *JMLR*, 3:1137–1155.

J. Bilmes, K. Asanovic, C. Chin, and J. Demmel. 1997. Using phipac to speed error back-propagation learning. *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, 5:4153.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proc. ACL'96*, pages 310–318, San Francisco.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proc. of ICML'08*, pages 160–167, New York, NY, USA. ACM.

Ahmed Emami and Lidia Mangu. 2007. Empirical study of neural network language models for Arabic speech recognition. In *Proc. ASRU'07*, pages 147–152, Kyoto. IEEE.

Geoffrey E. Hinton and Ruslan R. Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July.

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 *Proc. ACL'07*, pages 177–180, Prague, Czech Republic.

Philipp Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press.

Hong-Kwang Kuo, Lidia Mangu, Ahmad Emami, and Imed Zitouni. 2010. Morphological and syntactic features for arabic speech recognition. In *Proc. ICASSP 2010*.

Raymond Lau, Ronald Rosenfeld, and Salim Roukos. 1993. Adaptive language modeling using the maximum entropy principle. In *Proc HLT'93*, pages 108–113, Princeton, New Jersey.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proc. ICML '07*, pages 641–648, New York, NY, USA.

Andriy Mnih and Geoffrey E Hinton. 2008. A scalable hierarchical distributed language model. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, volume 21, pages 1081–1088.

Thomas R. Niesler. 1997. *Category-based statistical language models*. Ph.D. thesis, University of Cambridge.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. ACL'03*, pages 160–167, Sapporo, Japan.

Ilya Oparin, Ondřej Glembek, Lukáš Burget, and Jan Černocký. 2008. Morphological random forests for language modeling of inflectional languages. In *Proc. SLT'08*, pages 189–192.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. ACL'02*, pages 311–318, Philadelphia.

Ronald Rosenfeld. 1996. A maximum entropy approach to adaptive statistical language modeling. *Computer, Speech and Language*, 10:187–228.

Holger Schwenk and Jean-Luc Gauvain. 2002. Connectionist language modeling for large vocabulary continuous speech recognition. In *Proc. ICASSP*, pages 765–768, Orlando, FL.

Holger Schwenk, Daniel Déchelotte, and Jean-Luc Gauvain. 2006. Continuous space language models for statistical machine translation. In *Proc. COLING/ACL'06*, pages 723–730.

Holger Schwenk. 2007. Continuous space language models. *Comput. Speech Lang.*, 21(3):492–518.

Yeh W. Teh. 2006. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proc. of ACL'06*, pages 985–992, Sidney, Australia.

Peng Xu and Frederik Jelinek. 2004. Random forests in language modeling. In *Proceedings of EMNLP'2004*, pages 325–332, Barcelona, Spain.

Enhancing domain portability of Chinese segmentation model using chi-square statistics and bootstrapping

Baobao Chang, Dongxu Han
Institute of Computational Linguistics, Peking University
Key Laboratory of Computational Linguistics(Peking University), Ministry Education, China
Beijing, 100871, P.R.China
chbb@pku.edu.cn,hweibo@126.com

Abstract

Almost all Chinese language processing tasks involve word segmentation of the language input as their first steps, thus robust and reliable segmentation techniques are always required to make sure those tasks well-performed. In recent years, machine learning and sequence labeling models such as Conditional Random Fields (CRFs) are often used in segmenting Chinese texts. Compared with traditional lexicon-driven models, machine learned models achieve higher F-measure scores. But machine learned models heavily depend on training materials. Although they can effectively process texts from the same domain as the training texts, they perform relatively poorly when texts from new domains are to be processed. In this paper, we propose to use χ^2 statistics when training an SVM-HMM based segmentation model to improve its ability to recall OOV words and then use bootstrapping strategies to maintain its ability to recall IV words. Experiments show the approach proposed in this paper enhances the domain portability of the Chinese word segmentation model and prevents drastic decline in performance when processing texts across domains.

1 Introduction

Chinese word segmentation plays a fundamental role in Chinese language processing tasks, because almost all Chinese language processing tasks are assumed to work with segmented input. After intensive research for more than twenty years, the performance of Chinese segmentation made considerable progress. The bakeoff series hosted by the Chinese Information Processing Society (CIPS) and ACL SIGHAN shows that an F measure of 0.95 can be achieved in the closed test tracks, in which only specified training materials can be used in learning segmentation models[1].

Traditional word segmentation approaches are lexicon-driven (Liang, 1987) and assume predefined lexicons of Chinese words are available. Segmentation results are obtained by finding a best match between the input texts and the lexicons. Such lexicon-driven approaches can be rule-based, statistic-based or in some hybrid form.

Xue (2003) proposed a novel way of segmenting Chinese texts, and it views the Chinese word segmentation task as a character tagging task. According to Xue's approach, no predefined Chinese lexicons are required; a tagging model is learned by using manually segmented training texts. The model is then used to assign each character a tag indicating the position of this character within a word. Xue's approach has become the most popular approach to Chinese word segmentation for its high performance and unified way of dealing with out-of-vocabulary (OOV) issues. Most segmentation work began to follow this approach later. Major improvements in this line of research include: 1) More sophisticated learning models were introduced other than the maximum entropy model that Xue used, such as the conditional random fields (CRFs) model which fits the sequence tagging tasks much better than the maximum entropy model (Tseng et al.,2005). 2) More tags were in-

[1] http://www.sighan.org/bakeoff2005/data/results.php.htm

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 789–798,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

troduced, as Zhao et al. (2006) shows 6 tags are superior to 4 tags. 3) New feature templates were added, such as the templates that were used in representing numbers, dates, letters etc. (Low et al., 2005)

Character tagging approaches require manually segmented training texts to learn models usually in a supervised way. The performance is always evaluated on a test set from the same domain as the training set. Such evaluation does not reveal its ability to deal with domain variation. Actually, when test set is from other domains than the domain where training set is from, the learned model normally underperforms substantially.

One of the main reasons of such performance degradation lies in the model's ability to cope with OOV words. Actually, even when the test set has the same domain properties as the training set, the ability of the model to recall OOV words is still the main obstacle to achieve better performance of segmentation. However, when the test set is different with the training set in nature, the OOV recall normally drops much more substantially, and becomes much lower.

Apart from the supervised approach, Sun et al. (2004) proposed an unsupervised way of Chinese word segmentation. The approach did not use any predefined lexicons or segmented texts. A statistic named as *md*, combining the mutual information and *t* score, was proposed to measure whether a string of characters forms word. The unsupervised nature of the approach means good ability to deal with domain variation. However, the approach did not show a segmentation performance as good as that of the supervised approach. The approach was not evaluated in F measurement, but in accuracy of word break prediction. As their experiment showed, the approach successfully predicted 85.88% of the word breaks, which is much lower than that of the character tagging approach if in terms of F measurement.

Aiming at preventing the OOV recall from dropping sharply and still maintaining an overall performance as good as that of the state-of-art segmenter when working with heterogeneous test sets, we propose in this paper to use a semi-supervised way for Chinese word segmentation task. Specifically, we propose to use χ^2 statistics together with bootstrapping strategies to build Chinese word segmentation model. The experiment shows the approach can effectively promote the OOV recall and lead to a higher overall performance. In addition, instead of using the popular CRF model, we use another sequence labeling model in this paper --- the hidden Markov Support Vector Machines (SVM-HMM) Model (Altun et al., 2003). We just wish to show that there are alternatives other than CRF model to use and comparable results can be obtained.

Our work differs from the previous supervised work in its ability to cope with domain variation and differs from the previous unsupervised work in its much better overall segmentation performance.

The rest of the paper is organized as follows: In section 2, we give a brief introduction to the hidden Markov Support Vector Machines, on which we rely to build the segmentation model. In section 3, we list the segmentation tags and the basic feature templates we used in the paper. In section 4 we show how χ^2 statistics can be encoded as features to promote OOV recall. In section 5 we give the bootstrapping strategy. In section 6, we report the experiments and in section 7 we present our conclusions.

2 The hidden Markov support vector machines

The hidden Markov support vector machine (SVM-HMM) is actually a special case of the structural support vector machines proposed by Tsochantaridis et al. (2005). It is a powerful model to solve the structure predication problem. It differs from support vector machine in its ability to model complex structured problems and shares the max-margin training principles with support vector machines. The hidden Markov support vector machine model is inspired by the hidden Markov model and is an instance of structural support vector machine dedicated to solve sequence labeling learning, a problem that CRF model is assumed to solve. In the SVM-HMM model, the sequence labeling problem is modeled by learning a discriminant function *F*: *X*×*Y*→*R* over the pairs of input sequence and label sequence, thus the prediction of the label sequence can be derived by maximizing *F* over all possible label sequences for a specific given input sequence x.

$$f(\mathbf{x};\mathbf{w}) = \arg\max_{\mathbf{y}\in Y} F(\mathbf{x},\mathbf{y};\mathbf{w})$$

In the structural SVMs, *F* is assumed to be linear in some combined feature representation of the input sequence and the label sequence $\boldsymbol{\psi}(\mathbf{x},\mathbf{y})$, i.e.

$$F(\mathbf{x},\mathbf{y};\mathbf{w}) = \langle \mathbf{w}, \boldsymbol{\psi}(\mathbf{x},\mathbf{y}) \rangle$$

Where **w** denotes a parameter vector, for the SVM-HMMs, the discriminant function is defined as follows.

$$F(x,y;\mathbf{w}) = \sum_{t=1..T}\sum_{y\in\Sigma}\langle \overline{\mathbf{w}}_y, \mathbf{\Phi}(\mathbf{x}^t)\rangle \delta(y^t,y) + \eta \sum_{t=1..T-1}\sum_{y\in\Sigma}\sum_{y'\in\Sigma} \hat{\mathbf{w}}_{y,y'}\delta(y^t,y)\delta(y^{t+1},y')$$

Here $\mathbf{w} = (\overline{\mathbf{w}}, \hat{\mathbf{w}})$, $\mathbf{\Phi}(\mathbf{x}^t)$ is the vector of features of the input sequence. $\delta(y^t, y)$ is the Kronecker function, i.e.,

$$\delta(y^t,y) = \begin{cases} 1 & \text{if } y^t = y \\ 0 & \text{if } y^t \neq y \end{cases}$$

The first term of the discriminant function is used to model the interactions between input features and labels, and the second term is used to model interactions between nearby labels. $\eta > 0$ is a scaling factor which balances the two types of contributions. (Tsochantaridis et al., 2005)

Like SVMs, parameter vector **w** is learned with the maximum margin principle by using training data. To control the complexity of the training problem, the cutting plane method is used to solve the resulted constrained optimization problem. Thus only a small subset of constraints from the full-sized optimization is checked to ensure a sufficiently accurate solution. Roughly speaking, SVM-HMM differs from CRF in its principle of training, and both of them could be used to deal with sequence labeling problem like Chinese word segmentation.

3 The tag set and the basic feature templates

As in most other work on segmentation, we use a 4-tag tagset, that is S for the character being a single-character-word by itself, B for the character beginning a multi-character-word, E for the character ending a multi-character-word and M for a character occurring in the middle of a multi-character-word.

We use the following feature templates, as are widely used in most segmentation work:

(a) C_n (n = -2, -1, 0, 1, 2)
(b) C_nC_{n+1} (n = -2, -1, 0, 1)
(c) $C_{-1}C_{+1}$

Here C refers to a character; n refers to the position index relative to the current character. By setting the above feature templates, we actually set a 5-character window to extract features, the current character, 2 characters to its left and 2 characters to its right.

In addition, we also use the following feature templates to extract features representing the character type:

(d) T_n (n = -2, -1, 0, 1, 2)
(e) T_nT_{n+1} (n = -2, -1, 0, 1)
(f) $T_{-1}T_{+1}$

Here T refers to a character type, and its value can be digit, letter, punctuation or Chinese character. The type feature is important, for there are two versions of Arabic numbers, Latin alphabets and punctuations in the Chinese texts. This is because all three kinds of characters have their internal codes defined in ASCII table, but the Chinese encoding standard like GB18030 assigns them with other double-byte codes. This causes problems for model learning as we encounter in the experiment. The training data we adopt in this paper only use numbers, letters and punctuation of double-byte codes. But the test data use both the double-byte and single-byte codes. If the type features are not introduced, most of the numbers, letters and punctuation of single-byte can not be segmented correctly. The type feature establishes links between the two versions of codes, for both versions of a digit, a letter or punctuation share the same type feature value. Actually, the encoding problem could be alternatively solved by a character normalization process. That is the mapping all single-byte versions of digits, letters and punctuations in the test sets into their double-byte counterparts as in the training set. We use the type features here to avoid any changes to the test sets.

4 The χ2 statistic features

χ^2 test is one of hypothesis test methods, which can be used to test if two events co-occur just by chance or not. A lower χ^2 score normally means the two co-occurred events are independent; otherwise they are dependent on each other. χ^2 score is widely used in computational linguistics to extract collocations or terminologies. Unsupervised segmentation approach also mainly relies on mutual information and t-score to identify words in Chinese texts (Sun et al., 2004). Inspired by their work, we believe that χ^2 statistics could also be incorporated into supervised segmentation models to deal with the OOV issue. The idea is very straightforward. If two continuous characters in the test set have a higher χ^2 score, it is highly likely they form a word or are part of a word even they are not seen in the training set.

The χ^2 score of a character bigram (i.e. two continuous characters in the text) C_1C_2 can be computed by the following formula.

$$\chi^2(C_1,C_2)=\frac{n\times(a\times d-b\times c)^2}{(a+b)\times(a+c)\times(b+d)\times(c+d)}$$

Here,

a refers to all counts of bigram C_1C_2 in the text;

b refers to all counts of bigrams that C_1 oc-curs but C_2 does not;

c refers to all counts of bigrams that C_1 does not occur but C_2 occurs;

d refers to all counts of bigrams that both C_1 and C_2 do not occur.

n refers to total counts of all bigrams in the text, apparently, $n= a + b + c + d$.

We do the χ^2 statistics computation to the training set and the test set respectively. To make the χ^2 statistics from the training set and test set comparable, we normalize the χ^2 scores by the following formula.

$$\chi^2_{norm}(C_1,C_2)=\left\lfloor\frac{\chi^2(C_1,C_2)-\chi^2_{\min}}{\chi^2_{\max}-\chi^2_{\min}}\times 10\right\rfloor$$

To make the learned model sensitive to the χ^2 statistics, we then add two more feature templates as follows:

(g) X_nX_{n+1} (n = -2, -1, 0, 1)

(h) $X_{-1}X_{+1}$

The value of the feature X_nX_{n+1} is the normalized χ^2 score of the bigram C_nC_{n+1}. Note we also compute the normalized χ^2 score to bigram $C_{-1}C_{+1}$, which is to measure the association strength of two intervened characters.

By using the χ^2 features, statistics from the test set are introduced into segmentation model, and it makes the resulted model more aware of the test set and therefore more robust to test domains other than training domains.

Because the normalized χ^2 score is one of 11 possible values 0, 1, 2, …, 10, templates (g)-(h) generate 55 features in total.

All features generated from the templates (a)-(f) together with the 55 χ^2 features form the whole feature set. The training set and test set are then converted into their feature representations. The feature representation of the training set is then used to learn the model and the feature representation of the test set is then used for segmentation and evaluated by comparison with gold standard segmentation. The whole process is shown in Figure-1.

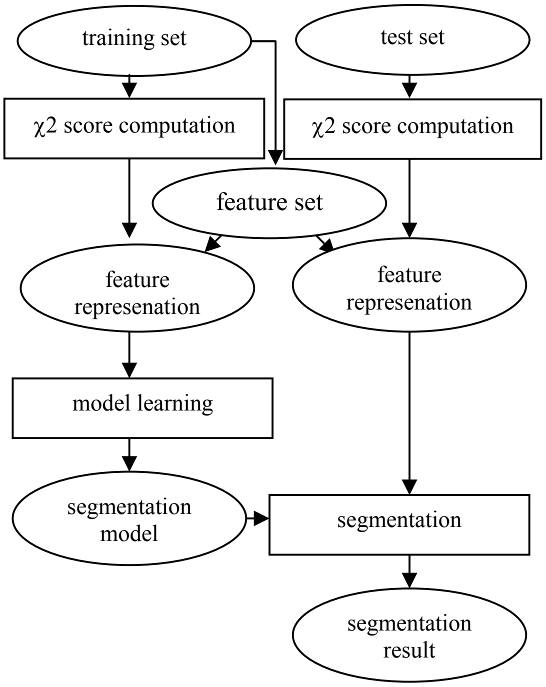


Figure-1. The workflow

By this way, an OOV word in the test set might be found by the segmentation model if the bigrams extracted from this word take higher χ^2 scores.

5 the bootstrapping strategy

The addition of the χ^2 features can be also problematic as we will see in the experiments. Even though it could promote the OOV recall significantly, it also leads to drops in in-vocabulary (IV) recall.

We are now in a dilemma. If we use χ^2 features, we get high OOV recall but a lower IV recall. If we do not use the χ^2 feature, we get a lower OOV recall but a high IV recall. To keep the IV recall from falling, we propose to use a bootstrapping method. Specifically, we choose to use both models with χ^2 features and without χ^2 features. We train two models firstly, one is χ^2-based and the other not. Then we do the segmentation for the test set with the two models simultaneously. Two segmentation results can be obtained. One result is produced by the χ^2-based model and has a high OOV recall. The other result is produced by the non-χ^2-based model and has a higher IV recall. Then we compare the two results and extract all sentences that have equal segmentations with the two models as the intersection of the two results. It is not difficult to understand that the intersection of the two results has both high OOV recall and high IV recall, if we also extract these sentences from the gold standard segmentation and perform evaluations. We then put the intersection results into the training set to form a new training set. By this new training set, we train again to get two new models, one χ^2-based and the other not. Then the two new models are used to segment the test set. Then we do again intersection to the two results and their common parts are again put into the training set. We repeat this process until a plausible result is obtained.

The whole process can be informally described as the following algorithm:

```
1. let training set T to be the original training set;
2. for I = 0 to K
    1) train the $\chi^2$-based model by using training set T;
    2) train the non-$\chi^2$-based model by using training set T;
    3) do segmentation by using the $\chi^2$-based model;
    4) do segmentation by using the non-$\chi^2$-based model;
    5) do intersection to the two segmentation results
    6) put the intersection results into the training set and get the enlarged training set T
3. train the non-$\chi^2$-based model using training set T, and take the output of this model as the final output;
4. end.
```

6 The experiments and discussions

6.1 On the training set and test set

For training the segmentation model, we use the training data provided by Peking University for bakeoff 2005[2]. The training set has about 1.1 million words in total. The PKU training data is actually consisted of all texts of the People's Daily newspaper in January of 1998. So the training data represents very formal written Chinese and mainly are news articles. A characteristic of the PKU data is that all Arabic numbers, Latin letters and punctuations in the data are all double-byte GB codes; there are no single-byte ASCII versions of these characters in the PKU training data.

We use three different test sets. The first one (denoted by A) is all texts of the People's Daily of February in 1998[3]. Its size and the genre of the texts are very similar to the training data. We use this test set to show how well the SVM-HMM can be used to model segmentation problem and the performance that a segmentation model achieves when applied to the texts from the same domain.

The second and the third test sets are set to test how well the segmentation model can apply to texts from other domains. The second test set (denoted by B) is from the literature domain and the third (denoted by C) from computer domain. We segmented them manually according to the guidelines of Peking University[4] to use as gold standard segmentations. The genres of the two test set are very different from the training set. There are even typos in the texts. In the computer test set, there are many numbers and English words. And most of the numbers and letters are single-byte ASCII codes.

The sizes and the OOV rates of the three test sets are shown in Table-1.

Table-1. Test sets statistics

test set	domain	word count	OOV rate
A	Newspaper	1,152,084	0.036
B	Literature	72,438	0.058
C	Computer	69,671	0.159

For all the experiments, we use the same evaluation measure as most of previous work on segmentation, that is the Recall(R), Precision(P), F measure (F=2PR/(P+R)), IV word recall and OOV word recall. In addition, we also evaluate all the test results with sentence accuracies (SA), which is the proportion of the correctly segmented sentences in the test set.

[2] can be download from http://www.sighan.org/bakeoff2005/

[3] The corpus can be licensed from Peking University.

[4] See http:// www.sighan.org/bakeoff2005/

Table-2. Performance of the SVM-HMM and CRF model

Models	P	R	F	Riv	Roov	SA
SVM-HMM	0.9566	0.9528	0.9547	0.9620	0.7041	0.5749
CRF	0.9541	0.9489	0.9515	0.9570	0.7185	0.5570

Table-3. Performance of the basic model

test set	P	R	F	Riv	Roov	SA
A	0.9566	0.9528	0.9547	0.9620	0.7041	0.5749
B	0.9135	0.9098	0.9116	0.9295	0.5916	0.4698
C	0.7561	0.8394	0.7956	0.9325	0.3487	0.2530

Table-4. Performance of the type sensitive model

test set	P	R	F	Riv	Roov	SA
A	0.9576	0.9522↓	0.9549	0.9610↓	0.7161	0.5766
B	0.9176	0.9095↓	0.9136	0.9273↓	0.6228	0.4832
C	0.9141	0.8975	0.9057	0.9381	0.6839	0.4287

Table-5. Performance of the χ^2-based model

test set	P	R	F	Riv	Roov	SA
A	0.9585	0.9518↓	0.9552	0.9602↓	0.7274	0.5736↓
B	0.9211	0.8971↓	0.9090↓	0.9104↓	0.6825	0.4648↓
C	0.9180	0.8895↓	0.9035↓	0.9209↓	0.7239	0.4204↓

Table-6. Performance of the bootstrapping model

test set	P	R	F	Riv	Roov	SA
B	0.9260	0.9183	0.9221	0.9329	0.6830	0.5120
C	0.9113↓	0.9268	0.9190	0.9482	0.8138	0.5039

6.1 SVM-HMM vs. CRF

To show how well the SVM-HMM model can be used to model segmentation tasks and its performance compared to that of CRF model, we use the training set to train two models, one with SVM-HMM and the other with CRF.

The implementations of SVM-HMM and CRF model we use in the paper can be found and downloaded respectively via Internet. [5]

To make the results comparable, we use the same feature templates, that is feature template (a)-(c). However, SVM-HMM takes interactions between nearby labels into the model, which means there is a label bigram feature template implicitly used in the SVM-HMM. So when training the CRF model we also use explicitly the label bigram feature template to model interactions between nearby labels[6].

For the SVM-HMM model, we set ε to 0.25. This is a parameter to control the accuracy of the solution of the optimization problem. We set C to half of the number of the sentences in the training data according to our understanding to the models. The C parameter is set to trade off the margin size and training error. For CRF model, we use all parameters to their default value. We do not do parameter optimizations to both models with respect their performances.

We use test set A to test both models. For both models, we use the same cutoff frequency to feature extraction. Only those features that are seen more than three times in texts are actually used in the models. The performances of the two models are shown in Table-2, which shows SVM-HMM can be used to model Chinese segmentation tasks

[5] http://www.cs.cornell.edu/People/tj/svm_light/svm_hmm.html, and http://sourceforge.net/projects/crfpp/

[6] specified by the B template as the toolkit requires.

and comparable results can be achieved like CRF model.

6.2 The baseline model

To test how well the segmentation model applies to other domain texts, we only use the SVM-HMM model with the same parameters as in section 6.1 and the same cutoff frequency.

For a baseline model, we only use feature templates (a)-(c), the performances of the basic model on the three test sets are shown in Table-3.

For the test set A, which is from the same domain as the training data, an F-score 0.95 is achieved.

For test set B and C, both are from different domains with the training data, the F-scores drop significantly. Especially the OOV recalls fall drastically, which means the model is very sensitive to the domain variation. Even the IV recalls fall significantly. This also shows the domain portability of the segmentation model is still an obstacle for the segmentation model to be used in cross-domain applications.

6.3 The type features

As we noted before, there are different encoding types for the Arabic numbers, Latin letters and punctuations. Especially, test set C is full of single-byte version of such numbers, letters and punctuations. The introduction of type features may improve performance of the model to the test set. Therefore, we use the feature tem-plates (a)-(f) to train a type sensitive model with the training data. This gives segmentation results shown in table-4. (The symbol ↓ means performance drop compared with a previous model)

As we can see, for test set A, the type features almost contribute nothing; the F-score has a very slight change. The IV recall even has a slight fall while the OOV recall rises a little.

For test set C, the type features bring about very significant improvement. The F-score rises from 0.7956 to 0.9057, and the OOV recall rises from 0.3487 to 0.6839. Different with the test set A, even the IV recall for test set C rises slightly. The reason of such a big improvement lies in that there are many single-byte digits, letters and punctuations in the texts.

Unlike test set C, there are not so many single-byte characters in test set B. Even though the OOV recall does rise significantly, the change in OOV recall for test set B is not as much as that for test set B. Type features contribute much to cross domain texts.

6.4 The $\chi2$-based model

Compared with OOV recall for test set A, the OOV recall for test set B and C are still lower. To promote the OOV recall, we use the feature templates (a)-(h) to train a χ^2-based model with the training data. This gives segmentation results shown in table-5.

As we see from table-5, the introduction of the χ^2 features does not improve the overall performance. Only F-score for test set A improves slightly, the other two get bad. But the OOV recall for the three test sets does improve, especially for test set B and C. The IV recalls for the three test sets drop, especially for test set B and C. That's why the F scores for test B and C drop.

6.5 Bootstrapping

To increase the OOV recall and prevent the IV recall from falling, we use the bootstrapping strategy in section 5.

We set K = 3 and run the algorithm shown in section 5. We just do the bootstrapping to test set B and C, because what we are concerned with in this paper is to improve the performance of the model to different domains. This gives results shown in Table-6. As we see in Table-6, almost all evaluation measurements get improved. Not only the OOV recall improves significantly, but also the IV recall improves compared with the type-sensitive model.

To illustrate how the bootstrapping strategy works, we also present the performance of the intermediate models on test set C in each pass of the bootstrapping in table-7 and table-8. Table-7 is results of the intermediate χ^2-based models for test set C. Table-8 is results of the intermediate non-χ^2-based models for test set C. Figure-2 illustrates changes in OOV recalls of both non-χ^2-based models and χ^2-based models as the bootstrapping algorithm advances for test set C. Figure-3 illustrates changes in IV re-calls of both non-χ^2-based models and χ^2-based models for test set C. As we can see from Figure-2 and Figure-3, the ability of non-χ^2-based model gets improved to the OOV

Table-7. Performance of the intermediate χ^2-based models for test set C

I	P	R	F	Riv	Roov	SA
0	0.9180	0.8895	0.9035	0.9209	0.7239	0.4204
1	0.9084	0.9186	0.9134	0.9387	0.8126	0.4762
2	0.9083	0.9187	0.9134	0.9386	0.8138	0.4822
3	0.9068	0.9208	0.9137	0.9412	0.8131	0.4816

Table-8. Performance of the intermediate non-χ^2-based models for test set C

I	P	R	F	Riv	Roov	SA
0	0.9141	0.8975	0.9057	0.9381	0.6839	0.4287
1	0.9070	0.9249	0.9159	0.9478	0.8044	0.4869
2	0.9093	0.9254	0.9173	0.9476	0.8087	0.4947
3	0.9111	0.9266	0.9188	0.9481	0.8133	0.5030
4	0.9113	0.9268	0.9190	0.9482	0.8138	0.5039

Table-9. Performance of the intersection of the intermediate χ^2-based model and non-χ^2-based model for test C

I	P	R	F	Riv	Roov	SA
0	0.9431	0.9539	0.9485	0.9664	0.8832	0.6783
1	0.9259	0.9434	0.9345	0.9609	0.8491	0.5992
2	0.9178	0.9379	0.9277	0.9582	0.8316	0.5724
3	0.9143	0.9347	0.9244	0.9559	0.8250	0.5616

recall of the χ^2-based model as the bootstrapping algorithm advances. The abilities to recall IV words of both models improve, and even the final IV recall of the χ^2-based model surpasses the IV recall of the type sensitive model shown in Table-3. (0.9412 vs. 0.9381). To save the space of the paper, we do not list all the intermediate results for test set B. We just show the changes in OOV recalls and IV recalls as illustrated in Figure-4 and Figure-5. One can see from Figure-4 and Figure-5, the bootstrapping strategy also works for test set B in a similar way as it works for test set C.

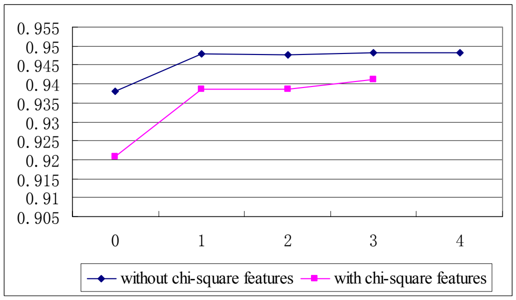


Figure-3 the Changes in IV recalls for test set C as boot-strapping algorithm advances

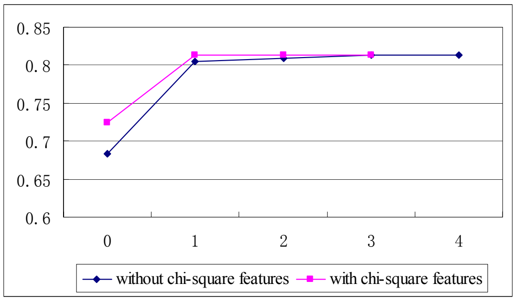


Figure-2 the Changes in OOV recalls for test set C as boot-strapping algorithm advances

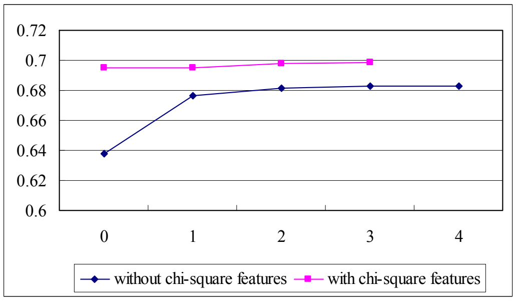


Figure-4 the Changes in OOV recalls for test set B as boot-strapping algorithm advances

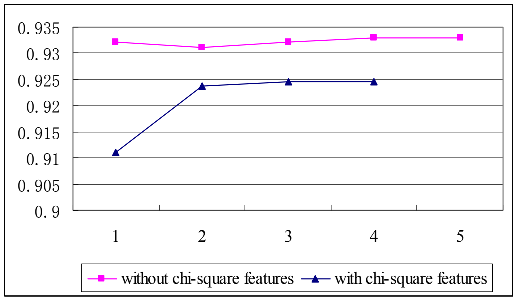


Figure-5 the Changes in IV recalls for test set B as boot-strapping algorithm advances

As we mentioned in section 5, the intersection of the results produced by χ^2-based model and non-χ^2-based model has both high OOV recall and high IV recall, that's the reason why bootstrapping strategy works. This can be seen from Table-9. However, as the algorithm progresses, both the OOV recall and IV recall of the intersection results fall, but are still higher than OOV recall and IV recall of the final results on the whole test set.

As we said before, we give also sentence accuracies of all segmentation models. With the χ^2 statistics and bootstrapping strategies, the sentence accuracy also rises. 2.8% more sentences on test set B and 7.5% more sentences on test set C are correctly segmented, compared with the type-sensitive model.

7 Conclusions

Sequence labeling models are widely used in Chinese word segmentation recently. High performance can be achieved when the test data is from the same domain as the training data. However, if the test data is assumed to be from other domains than the domain of the training data, the segmentation models always underperform substantially. To enhance the portability of the sequence labeling segmentation models to other domains, this paper proposes to use χ^2 statistics and bootstrapping strategy. The experiment shows the approach significantly increases both IV recall and OOV recall when processing texts from different domains.

We also show in this paper that hidden Markov support vector machine which is also a sequence labeling model like CRF can be used to model the Chinese word segmentation problem, by which high F-score results can be obtained like those of CRF model.

One concern to the bootstrapping approach in this paper is that it takes time to work with, which will make it difficult to be incorporated into language applications that need to responses in real time. However, we believe that such an approach can be used in offline contexts. For online use in a specified domain, one can first train models by using the approach in the paper with prepared raw texts from the specified domain and then use the final non-χ^2-based model to segment new texts of the same domain, since statistics of the target domain are more or less injected into the model by the iteration of bootstrapping.

Acknowledgments

This work is supported by National Natural Science Foundation of China under Grant No. 60975054 and National Social Science Foundation of China under Grant No. 06BYY048.

We would like to give thanks to Prof. Duan Huiming for her work in preparing the gold standard segmentation and to the anonymous reviewers for their comments to the paper.

References

Altun,Yasemin et al.,2003, Hidden Markov Support Vector Machines. Proceedings of the Twentieth Iternational Conference on Machine Learning (ICML-2003), Washington DC, 2003.

Gao, Jianfeng et al., 2005, Chinese Word Segmentation and Named Entity Recognition: A Pragmatic Approach, Computational Linguis-tics,Vol.31, No.4, pp531-574.

Huang, Changning et al. 2007, Chinese word segmentation: a decade review. Journal of Chinese Information Processing, Vol.21, NO.3,pp8–19.(in Chinese)

Liang, Nanyuan, 1987. ‘‘written Chinese text segmentation system--cdws”. Journal of Chinese Information Processing, Vol.2, NO.2, pp44–52.(in Chinese)

Low, Jin Kiat et al.,2005, A Maximum Entropy Approach to Chinese Word Segmentation. Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing, Jeju Island, Ko-rea,. pp161-164

Sun, Maosong et al., 2004, Chinese word segmentation without using dictionary based on unsupervised learning strategy. Chinese Journal of Computers. Vol.27, No.6, pp736-742. (in Chinese)

Tseng, Huihsin et al., 2005, A conditional random field word segmenter for SIGHAN 2005, Proceedings of the fourth SIGHAN workshop on Chinese language processing. Jeju Island, Korea. pp168-171

Tsochantaridis,Ioannis et al., 2005, Large Margin Methods for Structured and Interdependent Output Variables, Journal of Machine Learning Research (JMLR), No.6, pp1453-1484.

Xue, Nianwen, 2003, Chinese Word Segmentation as Character Tagging, Computational Linguistics and Chinese Language Processing. Vol.8, No.1, pp29-48.

Zhao, Hai et al., 2006, Effective tag set selection in Chinese word segmentation via conditional random field modeling, Proceedings of the 20th Pacific Asia Conference on language, Information and Computation (PACLIC-20), Wuhan, China, pp87-94

Latent-Descriptor Clustering for Unsupervised POS Induction

Michael Lamar
Department of Mathematics and Computer Science
Saint Louis University
220 N. Grand Blvd.
St.Louis, MO 63103, USA
mlamar@slu.edu

Yariv Maron
Gonda Brain Research Center
Bar-Ilan University
Ramat-Gan 52900, Israel
syarivm@yahoo.com

Elie Bienenstock
Division of Applied Mathematics
and Department of Neuroscience
Brown University
Providence, RI 02912, USA
elie@brown.edu

Abstract

We present a novel approach to distributional-only, fully unsupervised, POS tagging, based on an adaptation of the EM algorithm for the estimation of a Gaussian mixture. In this approach, which we call Latent-Descriptor Clustering (LDC), word types are clustered using a series of progressively more informative descriptor vectors. These descriptors, which are computed from the immediate left and right context of each word in the corpus, are updated based on the previous state of the cluster assignments. The LDC algorithm is simple and intuitive. Using standard evaluation criteria for unsupervised POS tagging, LDC shows a substantial improvement in performance over state-of-the-art methods, along with a several-fold reduction in computational cost.

1 Introduction

Part-of-speech (POS) tagging is a fundamental natural-language-processing problem, and POS tags are used as input to many important applications. While state-of-the-art supervised POS taggers are more than 97% accurate (Toutanova et al., 2003; Tsuruoka and Tsujii, 2005), unsupervised POS taggers continue to lag far behind. Several authors addressed this gap using limited supervision, such as a dictionary of tags for each word (Goldwater and Griffiths, 2007; Ravi and Knight, 2009), or a list of word prototypes for each tag (Haghighi and Klein, 2006). Even in light of all these advancements, there is still interest in a completely unsupervised method for POS induction for several reasons. First, most languages do not have a tag dictionary. Second, the preparation of such resources is error-prone. Third, while several widely used tag sets do exist, researchers do not agree upon any specific set of tags across languages or even within one language. Since tags are used as basic features for many important NLP applications (e.g. Headden et al. 2008), exploring new, statistically motivated, tag sets may also be useful. For these reasons, a fully unsupervised induction algorithm has both a practical and a theoretical value.

In the past decade, there has been a steady improvement on the completely unsupervised version of POS induction (Schütze, 1995; Clark, 2001; Clark, 2003; Johnson, 2007; Gao and Johnson, 2008; Graça et al., 2009; Abend et al., 2010; Lamar et al., 2010; Reichart et al., 2010; Berg-Kirkpatrick et al., 2010). Some of these methods use morphological cues (Clark, 2001; Clark, 2003; Abend et al., 2010; Reichart et al., 2010; Berg-Kirkpatrick et al., 2010), but all rely heavily on distributional information, *i.e.*, bi-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 799–809,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

gram statistics. Two recent papers advocate *non-disambiguating* models (Abend et al., 2010; Lamar et al., 2010): these assign the same tag to all tokens of a word type, rather than attempting to disambiguate words in context. Lamar et al. (2010) motivate this choice by showing how removing the disambiguation ability from a state-of-the-art disambiguating model results in increasing its accuracy.

In this paper, we present a novel approach to non-disambiguating, distributional-only, fully unsupervised, POS tagging. As in all non-disambiguating distributional approaches, the goal, loosely stated, is to assign the same tag to words whose contexts in the corpus are similar. Our approach, which we call Latent-Descriptor Clustering, or LDC, is an iterative algorithm, in the spirit of the K-means clustering algorithm and of the EM algorithm for the estimation of a mixture of Gaussians.

In conventional K-means clustering, one is given a collection of N objects described as N data points in an r-dimensional Euclidean space, and one seeks a clustering that minimizes the sum of intra-cluster squared distances, *i.e.*, the sum, over all data points, of the squared distance between that point and the centroid of its assigned cluster. In LDC, we similarly state our goal as one of finding a tagging, *i.e.*, cluster assignment, A, that minimizes the sum of intra-cluster squared distances. However, unlike in conventional K-means, the N objects to be clustered are themselves described by vectors—in a suitable manifold—that depend on the clustering A. We call these vectors *latent descriptors*.

Specifically, each object to be clustered, *i.e.*, each word type w, is described in terms of its *left-tag context* and *right-tag context*. These context vectors are the counts of the K different tags occurring, under tagging A, to the left and right of tokens of word type w in the corpus. We normalize each of these context vectors to unit length, producing, for each word type w, two points $L_A(w)$ and $R_A(w)$ on the $(K–1)$-dimensional unit sphere. The latent descriptor for w consists of the pair $(L_A(w), R_A(w))$—more details in Section 2.

A straightforward approach to this latent-descriptor K-means problem is to adapt the classical iterative K-means algorithm so as to handle the latent descriptors. Specifically, in each iteration, given the assignment A obtained from the previous iteration, one first computes the latent descriptors for all word types as defined above, and then proceeds in the usual way to update cluster centroids and to find a new assignment A to be used in the next iteration.

For reasons to be discussed in Section 5, we instead prefer a *soft-assignment* strategy, inspired from the EM algorithm for the estimation of a mixture of Gaussians. Thus, rather than the hard assignment A, we use a soft-assignment matrix P. P_{wk}, interpreted as the probability of assigning word w to cluster k, is, essentially, proportional to $\exp\{- d_{wk}^2/2\sigma^2\}$, where d_{wk} is the distance between the latent descriptor for w and the centroid, *i.e.*, Gaussian mean, for k. Unlike the Gaussian-mixture model however, we use the same mixture coefficient and the same Gaussian width for all k. Further, we let the Gaussian width σ decrease gradually during the iterative process. As is well-known, the EM algorithm for Gaussian mixtures reduces in the limit of small σ to the simpler K-means clustering algorithm. As a result, the last few iterations of LDC effectively implement the hard-assignment K-means-style algorithm outlined in the previous paragraph. The soft assignment used earlier in the process lends robustness to the algorithm.

The LDC approach is shown to yield substantial improvement over state-of-the-art methods for the problem of fully unsupervised, distributional only, POS tagging. The algorithm is conceptually simple and easy to implement, requiring less than 30 lines of Matlab code. It runs in a few seconds of computation time, as opposed to hours or days for the training of HMMs.

2 Notations and Statement of Problem

The LDC algorithm is best understood in the context of the latent-descriptor K-means optimization problem. In this section, we set up our notations and define this problem in detail. For simplicity, induced tags are henceforth referred to as *labels*, while *tags* will be reserved for the gold-standard tags, to be used later for evaluation.

Let W denote the set of word types $w_1,…,w_N$, and let T denote the set of labels, *i.e.*, induced tags. The sizes of these sets are $|W| = N$ and $|T| = K$. In the experiments presented in Section 4, N is 43,766 and K is either 50 or 17. For any word token t in the corpus, we denote the word type of t by $w(t)$. The frequency of word type w in the corpus is denoted $f(w)$; thus, $\Sigma_w f(w) = 1$.

For a word type w_1, the *left-word context* of w_1, $L(w_1)$, is defined as the N-dimensional vector whose n-th component is the number of bigrams, *i.e.*, pairs of consecutive tokens (t_{i-1}, t_i) in the corpus, such that $w(t_i) = w_1$ and $w(t_{i-1}) = n$. Similarly, we define the *right-word context* of w_1, $R(w_1)$, as the N-dimensional vector whose n-th component is the number of bigrams (t_i, t_{i+1}) such that $w(t_i) = w_1$ and $w(t_{i+1}) = n$. We let L (resp. R) be the $N \times N$ matrix whose w-th row is $L(w)$ (resp. $R(w)$).

S^{K-1} is the unit sphere in the K-dimensional Euclidean space $\mathbb{R}^K$. For any $x \in \mathbb{R}^K$, we denote by $\lambda(x)$ the projection of x on S^{K-1}, *i.e.*, $\lambda(x) = x/\|x\|$.

A labeling is a map A: $W \rightarrow T$. Given a labeling A, we define $\widetilde{L}_A(w_1)$, the *left-label context* of word type w_1, as the K-dimensional vector whose k-th component is the number of bigrams (t_{i-1}, t_i) in the corpus such that $w(t_i) = w_1$ and $A(w(t_{i-1})) = k$. We define the *left descriptor* of word type w as:

$$L_A(w) = \lambda(\widetilde{L}_A(w)).$$

We similarly define the right-label context of w_1, $\widetilde{R}_A(w_1)$, as the K-dimensional vector whose k-th component is the number of bigrams (t_i, t_{i+1}) such that $w(t_i) = w_1$ and $A(w(t_{i+1})) = k$, and we define the *right descriptor* of word type w as:

$$R_A(w) = \lambda(\widetilde{R}_A(w)).$$

In short, any labeling A defines two maps, L_A and R_A, each from W to S^{K-1}.

For any function $g(w)$ defined on W, $<g(w)>$ will be used to denote the average of $g(w)$ weighted by the frequency of word type w in the corpus:

$$<g(w)> = \Sigma_w f(w) g(w).$$

For any label k, we define:

$$C_L(k) = \lambda(< L_A(w): A(w) = k >).$$

Thus, $C_L(k)$ is the projection on S^{K-1} of the weighted average of the left descriptors of the word types labeled k. We sometimes refer to $C_L(k)$ as the left centroid of cluster k. Note that $C_L(k)$ depends on A in two ways, first in that the average is taken on words w such that $A(w) = k$, and second through the global dependency of L_A on A. We similarly define the right centroids:

$$C_R(k) = \lambda(< R_A(w): A(w) = k >).$$

Informally, we seek a labeling A such that, for any two word types w_1 and w_2 in W, w_1 and w_2 are labeled the same if and only if $L_A(w_1)$ and $L_A(w_2)$ are close to each other on S^{K-1} and so are $R_A(w_1)$ and $R_A(w_2)$. Formally, our goal is to find a labeling A that minimizes the objective function:

$$F(A) = <\|L_A(w) - C_L(A(w))\|^2 + \|R_A(w) - C_R(A(w))\|^2>.$$

Note that, just as in conventional K-means clustering, $F(A)$ is the sum of the intra-cluster squared distances. However, unlike conventional K-means clustering, the descriptors of the objects to be clustered depend themselves on the clustering. We accordingly refer to L_A and R_A as *latent descriptors*, and to the method described in the next section as **Latent-Descriptor Clustering**, or LDC.

Note, finally, that we do not seek the *global* minimum of $F(A)$. This global minimum, 0, is obtained by the trivial assignment that maps all word types to a unique label. Instead, we seek a minimum under the constraint that the labeling be non-trivial. As we shall see, this constraint need not be imposed explicitly: the iterative LDC algorithm, when suitably initialized and parameterized, converges to non-trivial local minima of $F(A)$—and these are shown to provide excellent taggers.

3 Methods

Recall that a mixture of Gaussians is a generative model for a random variable taking values in a Euclidean space $\mathbb{R}^r$. With K Gaussians, the model is parameterized by:

- K mixture parameters, *i.e.*, K non-negative numbers adding up to 1;
- K means, *i.e.*, K points $\mu_1,\ldots,\mu_K$ in $\mathbb{R}^r$;
- K variance-covariance $d\times d$ matrices.

The collection of all parameters defining the model is denoted by θ. EM is an iterative algorithm used to find a (local) maximizer of the likelihood of N observed data points $x_1,\ldots,x_N \in \mathbb{R}^r$. Each iteration of the algorithm includes an E phase and an M phase. The E phase consists of computing, based on the current θ, a probabilistic assignment of each of the N observations to the K Gaussian distributions. These probabilistic assignments form an $N\times K$ stochastic matrix P, *i.e.*, a matrix of non-negative numbers in which each row sums to 1. The M phase consists of updating the model parameters θ, based on the current assignments P. For more details, see, e.g., Bishop (2006).

The structure of the LDC algorithm is very similar to that of the EM algorithm. Thus, each iteration of LDC consists of an E phase and an M phase. As observations are replaced by latent descriptors, an iteration of LDC is best viewed as starting with the M phase. The M phase first starts by building a pair of latent-descriptor matrices L_P and R_P, from the soft assignments obtained in the previous iteration. Note that these descriptors are now indexed by P, the matrix of probabilistic assignments, rather than by hard assignments A as in the previous section.

L_P and R_P are obtained by a straightforward adaptation of the definition given in the previous section to the case of probabilistic assignments. Thus, the latent descriptors consist of the left-word and right-word contexts (recall that these are given by matrices L and R), mapped into left-label and right-label contexts through multiplication by the assignment matrix P, and scaled to unit length:

$$L_P = \lambda(LP)$$
$$R_P = \lambda(RP).$$

With these latent descriptors in hand, we proceed with the M phase of the algorithm as usual. Thus, the left mean $\mu^L{}_k$ for Gaussian k is the weighted average of the left latent descriptors $L_P(w)$, scaled to unit length. The weight used in this weighted average is $P_{wk}\times f(w)$ (remember that $f(w)$ is the frequency of word type w in the corpus). Note that the definition of the Gaussian mean $\mu^L{}_k$ parallels the definition of the cluster centroid $C_L(k)$ given in the previous section; if the assignment P happens to be a hard assignment, $\mu^L{}_k$ is actually identical to $C_L(k)$. The right Gaussian mean $\mu^R{}_k$ is computed in a similar fashion. As mentioned, we do not estimate any mixture coefficients or variance-covariance matrices.

The E phase of the iteration takes the latent descriptors and the Gaussian means, and computes a new $N\times K$ matrix of probabilistic assignments P. These new assignments are given by:

$$P_{wk} = \frac{1}{Z}\exp\{-[\| L_P(w) - \mu_k^L \|^2 + \| R_P(w) - \mu_k^R \|^2]/2\sigma^2\}$$

with Z a normalization constant such that $\sum_k P_{wk} = 1$. σ is a parameter of the model, which, as mentioned, is gradually decreased to enforce convergence of P to a hard assignment.

The description of the M phase given above does not apply to the first iteration, since the M phase uses P from the previous iteration. To initialize the algorithm, *i.e.*, create a set of left and right descriptor vectors in the M phase of the first iteration, we use the left-word and right-word contexts L and R. These matrices however are of very high dimension ($N\times N$), and thus sparse and noisy. We therefore reduce their dimensionality, using reduced-rank singular-value decomposition. This yields two $N\times r_1$ matrices, L_1 and R_1. A natural choice for r_1 is $r_1 = K$, and this was indeed used for $K = 17$. For $K = 50$, we also use $r_1 = 17$. The left and right descriptors for the first iteration are obtained by scaling each row of matrices L_1 and R_1 to unit length. The Gaussian centers $\mu^L{}_k$ and $\mu^R{}_k$, $k = 1,\ldots,K$, are set equal to the left and right descriptors of the K

most frequent words in the corpus. This completes the description of the LDC algorithm.[1]

While this algorithm is intuitive and simple, it does not easily lend itself to mathematical analysis; indeed there is no *a priori* guarantee that it will behave as desired. Even for the simpler, hard-assignment, K-means-style version of LDC outlined in the previous section, there is no equivalent to the statement—valid for the conventional K-means algorithm—that each iteration lowers the intra-cluster sum of squared distances $F(A)$; this is a mere consequence of the fact that the descriptors themselves are updated on each iteration. The soft-assignment version of LDC does not directly attempt to minimize $F(A)$, nor can it be viewed as likelihood maximization—as is EM for a Gaussian mixture—since the use of latent descriptors precludes the definition of a generative model for the data. This theoretical difficulty is compounded by the use of a variable σ.

Empirically however, as shown in the next section, we find that the LDC algorithm is very well behaved. Two simple tools will be used to aid in the description of the behavior of LDC.

The first tool is an objective function $G(P)$ that parallels the definition of $F(A)$ for hard assignments. For a probabilistic assignment P, we define $G(P)$ to be the weighted average, over all w and all k, of $\|L_P(w) - \mu^L_k\|^2 + \|R_P(w) - \mu^R_k\|^2$; the weight used in this average is $P_{wk}\times f(w)$, just as in the computation of the Gaussian means. Clearly, G is identical to F on any P that happens to be a hard assignment. Thus, G is actually an extension of the objective function F to soft assignments.

The second tool will allow us to compute a tagging accuracy for soft assignments. For this purpose, we simply create, for any probabilistic assignment P, the obvious labeling $A = A^*(P)$ that maps w to k with highest P_{wk}.

4 Results

In order to evaluate the performance of LDC, we apply it to the *Wall Street Journal* portion of the Penn Treebank corpus (1,173,766 tokens, all lower-case, resulting in $N = 43{,}766$ word types). We compare the induced labels with two gold-standard tagsets:

- **PTB45**, the standard 45-tag PTB tagset. When using PTB45 as the gold standard, models induce 50 labels, to allow comparison with Gao and Johnson (2008) and Lamar et al. (2010).

- **PTB17**, the PTB tagset coarse-grained to 17 tags (Smith and Eisner 2005). When using PTB17 as the gold standard, models induce 17 labels.

In order to compare the labels generated by the unsupervised model with the tags of each tagset, we use two map-based criteria:

- **MTO**: many-to-one tagging accuracy, *i.e.*, fraction of correctly-tagged tokens in the corpus under the so-called many-to-one mapping, which takes each induced tag to the gold-standard POS tag with which it co-occurs most frequently. This is the most prevalent metric in use for unsupervised POS tagging, and we find it the most reliable of all criteria currently in use. Accordingly, the study presented here emphasizes the use of MTO.

- **OTO**: best tagging accuracy achievable under a so-called one-to-one mapping, *i.e.*, a mapping such that at most one induced tag is sent to any POS tag. The optimal one-to-one mapping is found through the Hungarian algorithm[2].

[1] The LDC code, including tagging accuracy evaluation, is available at http://www.dam.brown.edu/people/elie/code/.

[2] Code by Markus Beuhren is available at http://www.mathworks.com/matlabcentral/fileexchange/6543-functions-for-the-rectangular-assignment-problem

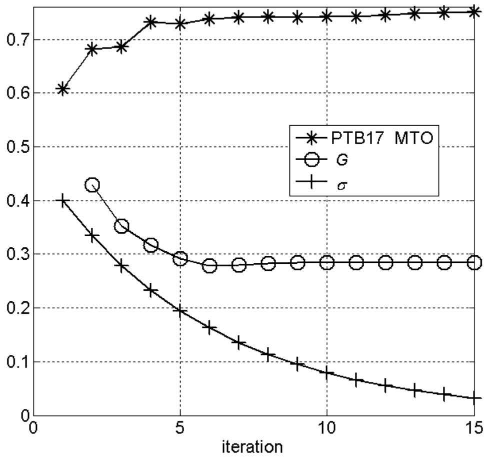


Figure 1: Convergence of LDC with $K = 17$. Bottom curve: σ -schedule, *i.e.*, sequence of Gaussian widths employed. Middle curve: Objective function $G(P)$ (see Section 3). Top curve: Many-to-one tagging accuracy of labeling $A^*(P)$, evaluated against PTB17.

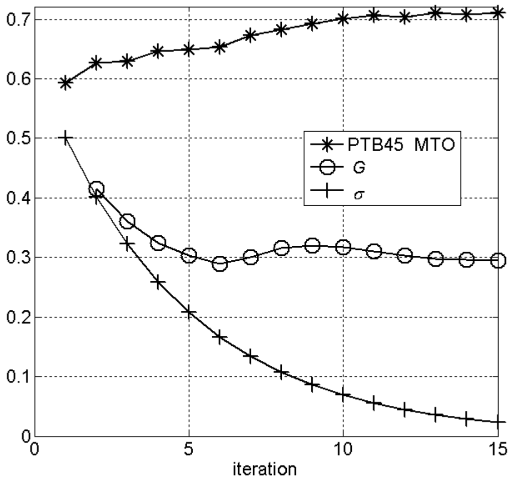


Figure 2: Same as Figure 1 but with $K = 50$. Top curve shows the MTO accuracy of the labeling evaluated against PTB45.

Figures 1 and 2 show the behavior of the LDC algorithm for K = 17 and K = 50 respectively. From the G curves as well as from the MTO scoring curves (using the labeling $A^*(P)$ defined at the end of Section 3), it is clear that the algorithm converges. The figures show only the first 15 iterations, as little change is observed after that. The schedule of the σ parameter was given the simple form $\sigma(t) = \sigma_1 \exp\{-c(t-1)\}$, t = 1,2,…, and the parameters σ_1 and c were adjusted so as to get the best MTO accuracy. With the σ-schedules used in these experiments, P typically converges to a hard assignment in about 45 iterations, σ being then 10^{-5}.

While the objective function $G(P)$ mostly decreases, it does show a hump for K = 50 around iteration 9. This may be due to the use of latent descriptors, or of a variable σ, or both. The MTO score sometimes decreases by a small fraction of a percent, after having reached its peak around the 15th iteration.

Note that we start σ at 0.4 for K = 17, and at 0.5 for K = 50. Although we chose two slightly different σ schedules for the two tagsets in order to achieve optimal performance on each tagset, an identical sequence of σ can be used for both with only a 1% drop in PTB17 score.

As the width of the Gaussians narrows, each vector is steadily pushed toward a single choice of cluster. This forced choice, in turn, produces more coherent descriptor vectors for all word types, and yields a steady increase in tagging accuracy.

Criterion	Model	PTB17	PTB45
MTO	LDC	**0.751**	**0.708**
	SVD2	0.740	0.658
	HMM-EM	0.647	0.621
	HMM-VB	0.637	0.605
	HMM-GS	0.674	0.660
OTO	LDC	**0.593**	0.483
	SVD2	0.541	0.473
	HMM-EM	0.431	0.405
	HMM-VB	0.514	0.461
	HMM-GS	0.466	**0.499**

Table 1. Tagging accuracy comparison between several models for two tagsets and two mapping criteria. Note that LDC significantly outperforms all HMMs (Gao and Johnson, 2008) in every case except PTB45 under the OTO mapping. LDC also outperforms SVD2 (Lamar et al., 2010) in all cases.

Table 1 compares the tagging accuracy of LDC with several recent models of Gao and Johnson (2008) and Lamar et al. (2010).

The LDC results shown in the top half of the table, which uses the MTO criterion, were obtained with the same σ-schedules as used in Figures 1 and 2. Note that the LDC algorithm is deterministic. However, the randomness in the sparse-matrix implementation of reduced-rank SVD used in the initialization step causes a small variability in performance (the standard deviation of the MTO score is 0.0004 for PTB17 and 0.003 for PTB45). The LDC results reported are averages over 20 runs. Each run was halted at iteration 15, and the score reported uses the labeling $A^*(P)$ defined at the end of Section 3.

The LDC results shown in the bottom half of the table, which uses the OTO criterion, were obtained with a variant of the LDC algorithm, in which the M phase estimates not only the Gaussian means but also the mixture coefficients. Also, different σ-schedules were used,[3]

For both PTB17 and PTB45, and under both criteria, LDC's performance nearly matches or exceeds (often by a large margin) the results achieved by the other models. We find the large increase achieved by LDC in the MTO performance under the PTB45 tagset particularly compelling. It should be noted that Abend et al. (2010) report 71.6% MTO accuracy for PTB45, but they treat all punctuation tags differently in their evaluation and therefore these results cannot be directly compared. Berg-Kirkpatrick et al. (2010) report 75.5% MTO accuracy for PTB45 by incorporating other features such as morphology; Table 1 is limited to distributional-only methods.

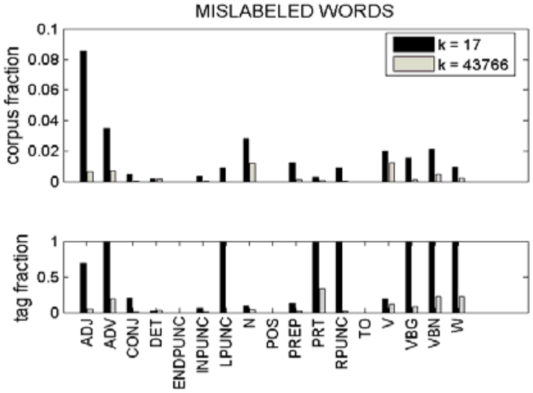


Figure 3: Mislabeled words per tag, using the PTB17 tagset. Black bars indicate mislabeled words when 17 clusters are used. Gray bars indicate words that continue to be mislabeled even when every word type is free to choose its own label, as if each type were in its own cluster—which defines the theoretically best possible non-disambiguating model. Top: fraction of the corpus mislabeled, broken down by gold tags. Bottom: fraction of tokens of each tag that are mislabeled. Many of the infrequent tags are 100% mislabeled because no induced label is mapped to these tags under MTO.

Figure 3 demonstrates the mistakes made by LDC under the MTO mapping. From the top graph, it is clear that the majority of the missed tokens are open-class words – most notably adjectives and adverbs. Over 8% of the tokens in the corpus are mislabeled adjectives – roughly one-third of all total mislabeled tokens (25.8%). Furthermore, the corresponding bar in the bottom graph indicates that over half of the adjectives are labeled incorrectly. Similarly, nearly 4% of the mislabeled tokens are adverbs, but *every* adverb in the corpus is mislabeled because no label is mapped to this tag – a common oc-

[3] All details are included in the code available at http://www.dam.brown.edu/people/elie/code/.

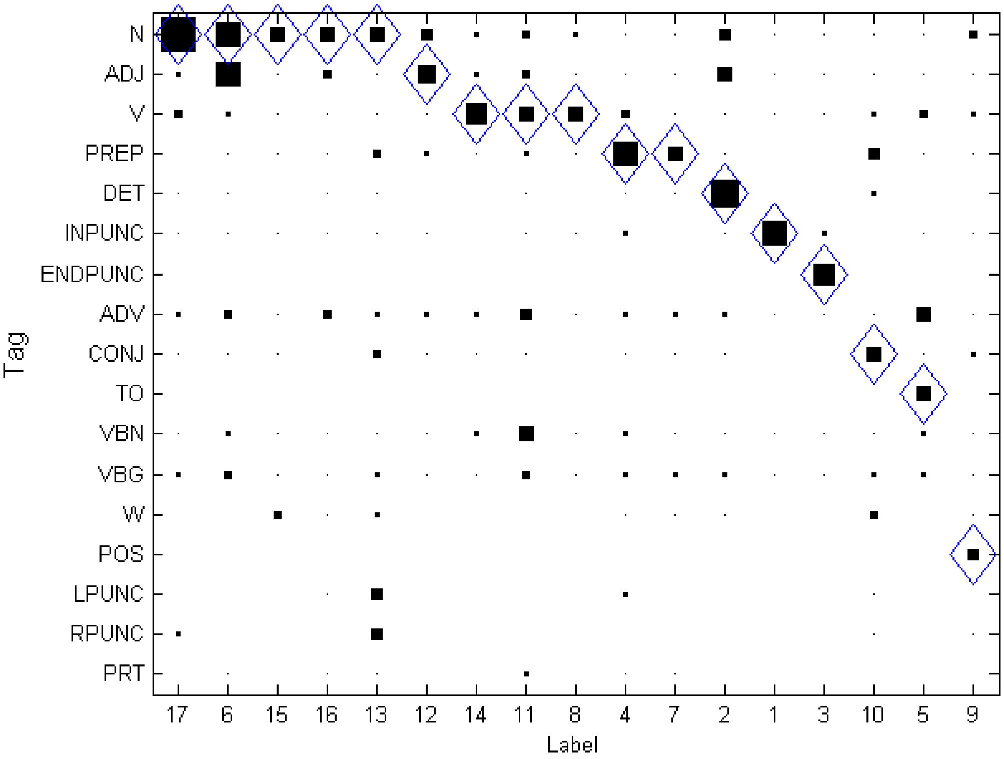


Figure 4: The confusion matrix for LDC's labeling under PTB17. The area of a black square indicates the number of tokens in each element of the confusion matrix. The diamonds indicate the induced tag under the MTO mapping. Several labels are mapped to N (Noun), and one of these labels causes appreciable confusion between nouns and adjectives. Because multiple labels are dedicated to a single tag (N, V and PREP), several tags (in this case 7) are left with no label.

currence under MTO, shared by seven of the seventeen tags.

To further illuminate the errors made by LDC, we construct the confusion matrix (figure 4). Element (*i,j*) of this matrix stores the fraction of all tokens of POS tag *i* that are given label *j* by the model. In a perfect labeling, exactly one element of each row and each column would be non-zero. As illustrated in figure 4, the confusion matrices produced by LDC are far from perfect. LDC consistently splits the Nouns into several labels and often confuses Nouns and Adjectives under a single label. These types of mistakes have been observed as well in models that use supervision (Haghighi and Klein, 2006).

5 Discussion

When devising a model for unsupervised POS induction, one challenge is to choose a model of adequate complexity, this choice being related to the bias-variance dilemma ubiquitous in statistical estimation problems. While large datasets are available, they are typically not large enough to allow efficient unsupervised learnability in models that are powerful enough to capture complex features of natural languages. Ambiguity is one of these features. Here we propose a new approach to this set of issues: start with a model that explicitly entertains ambiguity, and gradually constrain it so that it eventually converges to an unambiguous tagger.

Thus, although the algorithm uses probabilistic assignments, of Gaussian-mixture type, the goal is the construction of hard assignments. By requiring the Gaussians to be isotropic with uniform width and by allowing that width to shrink to zero, the algorithm forces the soft assignments to converge to a set of hard assignments. Based on its performance, this simulated-annealing-like approach appears to provide a good compromise in the choice of model complexity.

LDC bears some similarities with the algorithm of Ney, Essen and Kneser (1994), further implemented, with extensions, by Clark (2003). Both models use an iterative approach to minimize an objective function, and both initialize with frequent words. However, the model of Ney et al. is, in essence, an HMM where each word type is constrained to belong to a single class (i.e., in HMM terminology, be emitted by a single hidden state). Accordingly, the objective function is the data likelihood under this constrained HMM. This takes into account only the rightward transition probabilities. Our approach is conceptually rather different from an HMM. It is more similar to the approach of Schütze (1995) and Lamar et al. (2010), where each word type is mapped into a descriptor vector derived from its left and right tag contexts. Accordingly, the objective function is that of the *K*-means clustering problem, namely a sum of intra-cluster squared distances. This objective function, unlike the likelihood under an HMM, takes into account both left and right contexts. It also makes use in a crucial way of cluster centroids (or Gaussian means), a notion that has no counterpart in the HMM approach. We note that LDC achieves much better results (by about 10%) than a recent implementation of the Ney et al. approach (Reichart et al. 2010).

The only parameters in LDC are the two parameters used to define the σ schedule, and r_1 used in the first iteration. Performance was generally found to degrade gracefully with changes in these parameters away from their optimal values. When σ was made too large in the first few iterations, it was found that the algorithm converges to the trivial minimum of the objective function $F(A)$, which maps all word types to a unique label (see section 2). An alternative would be to estimate the variance for each Gaussian separately, as is usually done in EM for Gaussian mixtures. This would not necessarily preclude the use of an iteration-dependent scaling factor, which would achieve the goal of gradually forcing the tagging to become deterministic. Investigating this and related options is left for future work.

Reduced-rank SVD is used in the initialization of the descriptor vectors, for the optimization to get off the ground. The details of this initialization step do not seem to be too critical, as witnessed by robustness against many parameter changes. For instance, using only the 400 most frequent words in the corpus—instead of all words—in the construction of the left-word and right-word context vectors in iteration 1 causes no appreciable change in performance.

The probabilistic-assignment algorithm was found to be much more robust against parameter changes than the hard-assignment version of LDC, which parallels the classical *K*-means clustering algorithm (see Section 1). We experimented with this hard-assignment latent-descriptor clustering algorithm (data not shown), and found that a number of additional devices were necessary in order to make it work properly. In particular, we found it necessary to use reduced-rank SVD on each iteration of the algorithm—as opposed to just the first iteration in the version presented here—and to gradually increase the rank *r*. Further, we found it necessary to include only the most frequent words at the beginning, and only gradually incorporate rare words in the algorithm. Both of these devices require fine tuning. Provided they are indeed appropriately tuned, the same level of performance as in the probabilistic-assignment version could be achieved. However, as mentioned, the behavior is much less robust with hard clustering.

Central to the success of LDC is the dynamic interplay between the progressively harder cluster assignments and the updated latent descriptor vectors. We operate under the assumption that if all word types were labeled optimally, words that share a label should have similar descriptor vectors arising from this optimal labeling. These similar vectors would continue to be clustered together, producing a stable equilibrium in the dynamic process. The LDC algorithm demonstrates that, despite starting far from this optimal labeling, the alternation between vector updates and assignment updates is able to produce steadily improving clusters, as seen by the steady increase of tagging accuracy.

We envision the possibility of extending this approach in several ways. It is a relatively simple matter to extend the descriptor vectors to include context outside the nearest neighbors, which may well improve performance. In view of the computational efficiency of LDC, which runs in under one minute on a desktop PC, the added computational burden of working with the extended context is not likely to be prohibitive. LDC could also be extended to include morphological or other features, rather than relying exclusively on context. Again, we would anticipate a corresponding increase in accuracy from this additional linguistic information.

References

Omri Abend, Roi Reichart and Ari Rappoport. Improved Unsupervised POS Induction through Prototype Discovery. 2010. In *Proceedings of the 48th Annual Meeting of the ACL.*

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer-Verlag, New York, LLC.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless Unsupervised Learning with Features. In *proceedings of NAACL 2010.*

Alexander Clark. 2001. The unsupervised induction of stochastic context-free grammars using distributional clustering. In *CoNLL.*

Alexander Clark. 2003. Combining distributional and morphological information for part of speech induction. In *10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 59–66.

Jianfeng Gao and Mark Johnson. 2008. A comparison of bayesian estimators for unsupervised Hidden Markov Model POS taggers. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 344–352.

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.

João V. Graça, Kuzman Ganchev, Ben Taskar, and Fernando Pereira. 2009. Posterior vs. Parameter Sparsity in Latent Variable Models. *Neural Information Processing Systems Conference* (NIPS).

Michael Lamar, Yariv Maron, Mark Johnson, Elie Bienenstock. 2010. SVD and Clustering for Unsupervised POS Tagging. In *Proceedings of the 48th Annual Meeting of the ACL.*

Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 320–327, New York City, USA, June. Association for Computational Linguistics.

William P. Headden, David McClosky, and Eugene Charniak. 2008. Evaluating unsupervised part-of-speech tagging for grammar induction. In *Proceedings of the International Conference on Computational Linguistics* (*COLING '08).*

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.

Hermann Ney, Ute Essen, and Reinhard Kneser. 1994. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech and Language*, 8, 1-38.

Roi Reichart, Raanan Fattal and Ari Rappoport. 2010. Improved Unsupervised POS Induction Using Intrinsic Clustering Quality and a Zipfian Constraint. *CoNLL.*

Sujith Ravi and Kevin Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP,* pages 504–512.

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.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual*

Meeting of the Association for Computational Linguistics (ACL'05), pages 354–362.

Kristina Toutanova, Dan Klein, Christopher D. Manning and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL* 2003, pages 252-259.

Yoshimasa Tsuruoka and Jun'ichi Tsujii. 2005. Bidirectional Inference with the Easiest-First Strategy for Tagging Sequence Data. In *Proceedings of HLT/EMNLP*, pp. 467-474.

A Probabilistic Morphological Analyzer for Syriac

Peter McClanahan, George Busby, Robbie Haertel, Kristian Heal †,
Deryle Lonsdale‡, Kevin Seppi, Eric Ringger
Department of Computer Science, ‡Department of Linguistics,
†Center for the Preservation of Ancient Religious Texts (CPART)
Brigham Young University
Provo, Utah 84604 USA
http://nlp.cs.byu.edu/

Abstract

We define a probabilistic morphological analyzer using a data-driven approach for Syriac in order to facilitate the creation of an annotated corpus. Syriac is an under-resourced Semitic language for which there are no available language tools such as morphological analyzers. We introduce novel probabilistic models for segmentation, dictionary linkage, and morphological tagging and connect them in a pipeline to create a probabilistic morphological analyzer requiring only labeled data. We explore the performance of models with varying amounts of training data and find that with about 34,500 labeled tokens, we can outperform a reasonable baseline trained on over 99,000 tokens and achieve an accuracy of just over 80%. When trained on all available training data, our joint model achieves 86.47% accuracy, a 29.7% reduction in error rate over the baseline.

1 Introduction

Our objective is to facilitate the annotation of a large corpus of classical Syriac (referred to simply as “Syriac” throughout the remainder of this work). Syriac is an under-resourced Western Semitic language of the Christian Near East and a dialect of Aramaic. It is currently employed almost entirely as a liturgical language but was a true spoken language up until the eighth century, during which time many prolific authors wrote in Syriac. Even today there are texts still being composed in or translated into Syriac. By automatically annotating these texts with linguistically useful information, we will facilitate systematic study by scholars of Syriac, the Near East, and Eastern Christianity. Furthermore, languages that are linguistically similar to Syriac (e.g., Arabic and Hebrew) may benefit from the methodology presented here.

Our desired annotations include morphological segmentation, links to dictionary entries, and morphological attributes. Typically, annotations of this kind are made with the assistance of language tools, such as morphological analyzers, segmenters, or part-of-speech (POS) taggers. Such tools do not exist for Syriac, but some labeled data does exist: Kiraz (1994) compiled an annotated version of the Peshitta New Testament (1920) and a concordance thereof. We aim to replicate this kind of annotation on a much larger scale with more modern tools, building up from the labeled New Testament data, our only resource. Motivated by this state of affairs, our learning and annotation framework requires only labeled data.

We approach the problem of Syriac morphological annotation by creating five probabilistic sub-models that can be trained in a supervised fashion and combined in a joint model of morphological annotation. We introduce novel algorithms for segmentation, dictionary linkage, and morphological tagging. We then combine these sub-models into a joint n-best pipeline. This joint model outperforms a strong, though naïve, baseline for all amounts of training data over about 9,900 word tokens.

1.1 Syriac Background

Since Syriac is an abjad, its writing system does not require vowels. As a dialect of Aramaic, it is an inflected language with a templatic (non-concatenative) morphology, based on a system of triliteral consonantal roots, with prefixes, suffixes, infixes, and enclitic particles. Syriac is written from

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 810–820,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

right to left. For the purposes of this work, all Syriac is transliterated according to the Kiraz (1994) transliteration[1] and is written left-to-right whenever transliterated; the Syriac appearing in the Serto script in this paper is shown right-to-left.

Since there is no standardized nomenclature for the parts of a Syriac word, we define the following terms to facilitate the definitions of segmentation, dictionary linkage, and morphological tagging:

- word token - contiguous characters delimited by whitespace and/or punctuation
- stem - an inflected form of the baseform and the main part of the word to which prefixes and suffixes can be attached; the affixes do not inflect the stem but include prepositions, object suffixes, and enclitic pronouns
- baseform - the dictionary citation form; also known as a lexeme or lemma
- root - the form from which the baseform is derived

To clarify, we will use an example word token ܠܡܠܟܟܘܢ, *LMLCCON*, which means "to your (masculine plural) king". For this word, the stem is ܡܠܟ, *MLC*; the baseform is ܡܠܟܐ, *MLCA* "king"; and the root is ܡܠܟ, *MLC*. To clarify, note that the word token (including the stem) can be spoken and written with vowels as diacritics; however, since the vowels are not written in common practice and since most text does not include them, this work omits any indication of vowels. Furthermore, the stem is an inflected baseform and does not necessarily form a word on its own. Also, the (unvocalized) stem and root are not necessarily identical. In Syriac, the same root ܡܠܟ, *MLC* is the foundation for other words such as promise, counsel, deliberate, reign, queen, kingdom, and realm.

1.2 Sub-tasks

Segmentation, or tokenization as it is sometimes called (e.g., Habash and Rambow, 2007), is the process of dividing a word token into its prefix(es) (if any), a stem, and a suffix (if any). For Syriac, each word token consists of exactly one stem, from zero to three prefixes, and zero or one suffix. Each prefix is exactly one character in length. Segmentation does not include the process of parsing the stem for its inflectional morphology; that step is handled separately in subsequent processes described below. While segmenting a Syriac word, we can handle all prefixes as a single unit. It is trivial to segment a prefix cluster into its individual prefixes (one character per prefix). Suffixes may be multiple characters in length and encode the morphological attributes of the suffix itself (not of the stem); the suffix usually encodes the object of the stem and has its own grammatical attributes, which we list later. As an example of token segmentation, for the word token ܠܡܠܟܟܘܢ, *LMLCCON*, the prefix is ܠ, *L* "to", the stem is ܡܠܟ, *MLC* "king", and the suffix is ܟܘܢ, *CON* "(masculine plural) your".

Dictionary linkage is the process of linking a stem to its associated baseform and root. In most Syriac dictionaries, all headwords are either baseforms or roots, and for a given word these are the only relevant entries in the dictionary. Each Syriac stem is derived from a baseform, and each baseform is derived from a root. There is ambiguity in this correspondence which can be caused by, among other things, homographic stems generated from different roots or even from homographic roots. As such, linkage may be thought of as two separate processes: (1) baseform linkage, where the stem is mapped to its most likely baseform; and (2) root linkage, where the baseform is mapped to its most likely root. For our example ܠܡܠܟܟܘܢ, *LMLCCON*, baseform linkage would map stem ܡܠܟ, *MLC* to baseform ܡܠܟܐ, *MLCA*, and root linkage would map baseform ܡܠܟܐ, *MLCA* to root ܡܠܟ, *MLC*.

Morphological tagging is the process of labeling each word token with its morphological attributes. Morphological tagging may be thought of as two separate tagging tasks: (1) tagging the stem and (2) tagging the suffix. For Syriac, scholars have defined for this task a set of morphological attributes consisting of twelve attributes for the stem and four attributes for the suffix. The attributes for the stem are as follows: grammatical category, verb conjugation, aspect, state, number, person, gender, pronoun type, demonstrative category, noun type, numeral type, and participle type. The morphological

[1] According to this transliteration all capital letters including *A* (ܐ, olaph) and *O* (ܘ, waw) are consonants. Additionally, the semi-colon (*;*), representing (ܝ, yod), is also a consonant.

Attribute	Value
Grammatical Category	noun
Verb Conjugation	N/A
Aspect	N/A
State	emphatic
Number	singular
Person	N/A
Gender	masculine
Pronoun Type	N/A
Demonstrative Category	N/A
Noun Type	common
Numeral Type	N/A
Participle Type	N/A

Table 1: The values for the morphological attributes of the stem ܡܠܟ, *MLC*, “king”.

Attribute	Value
Gender	masculine
Person	second
Number	plural
Contraction	normal suffix

Table 2: The values for the morphological attributes of the suffix ܟܘܢ, *CON*, “(masculine plural) your”.

attributes for the suffix are gender, person, number, and contraction. The suffix contraction attribute encodes whether the suffix is normal or contracted, a phonological process involving the attachment of an enclitic pronoun to a participle. These morphological attributes were heavily influenced by those used by Kiraz (1994), but were streamlined in order to focus directly on grammatical function. During morphological tagging, each stem is labeled for each of the stem attributes, and each suffix is labeled for each of the suffix attributes. For a given grammatical category (or POS), only a subset of the morphological attributes is applicable. For those morphological attributes (both of the stem and of the suffix) that do not apply, the correct label is “N/A” (not applicable). Tables 1 and 2 show the correct stem and suffix tags for the word ܠܡܠܟܟܘܢ, *LMLCCON*.

The remainder of the paper will proceed as follows: Section 3 outlines our approach. In Section 4, we describe our experimental setup; we present results in Section 5. Section 6 contrasts previous work with our approach. Finally, in Section 7 we briefly conclude and offer directions for future work.

2 The Syromorph Approach

Since lack language tools, we focus on automatically annotating Syriac text in a data-driven fashion based on the labeled data we have available. Since segmentation, linkage, and morphological tagging are not mutually independent tasks, we desire models for the sub-tasks to influence each other. To accommodate these requirements, we use a joint pipeline model (Finkel et al., 2006). In this section, we will first discuss this joint pipeline model, which we call **syromorph**. We then examine each of the individual sub-models.

2.1 Joint Pipeline Model

Our approach is to create a joint pipeline model consisting of a segmenter, a baseform linker, a root linker, a suffix tagger, and a stem tagger. Figure 1 shows the dependencies among the sub-models in the pipeline for a single word. Each sub-model (oval) has access to the data and predictions (rectangles) indicated by the arrows. For example, for a given word, the stem tagger has access to the previously predicted stem, baseform, root, and suffix tag. The baseform linker has access to the segmentation, most importantly the stem.

The training of **syromorph** is straightforward. Each of the individual sub-models is trained separately on the true labeled data. Features are extracted from the local context in the sentence. The local context consists first of predictions for the entire sentence from earlier sub-tasks (those sub-tasks upon which the sub-task in question depends). We created the dependencies shown in Figure 1 taking into account the difficulty of the tasks and natural dependencies in the language. In addition to the predictions for the entire sentence from previous sub-tasks, the local context also includes the previous *o* tags of the current sub-task, as the standard order *o* Markov model does. For example, when the stem tagger is being trained on a particular sentence, the local context consists of the words in the sentence, the predicted segmentation, baseform, root, and suffix tags for each word in the sentence, and additionally the labels for the previous *o* stems. To further elaborate

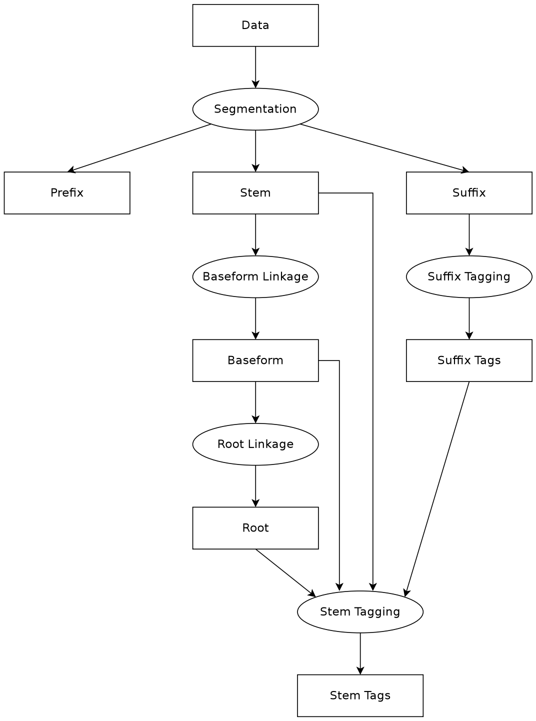


Figure 1: The **syromorph** model. Each rectangle is an input or output and each oval is a process employing a sub-model.

on the example, since features are extracted from the local context, for stem tagging we extract features such as current stem, previous stem, current baseform, previous baseform, current root, previous root, current suffix tags, and previous suffix tags. (Here, "previous" refers to labels on the immediately preceding word token.)

2.2 Segmentation

The **syromorph** segmentation model is a hybrid word- and consonant-level model, based on the model of Haertel et al. (2010) for data-driven diacritization. Each of our probabilistic sequence models is a maximum entropy Markov model (MEMM). Haertel et al. (2010) showed that the distribution over labels is different for known and words and rare words. In this work, we only consider words not seen in training (i.e., "unknown") to be rare. Following Haertel et al.'s (2010) model, a separate model is trained for each word type seen in training with the intent of choosing the best segmentation given that word. This approach is closely related to the idea of ambiguity classes mentioned in Hajič and Hladká (1998).

To handle unknown words, we back off to a consonant-level model. Our consonant-level segmentation model uses the notion of BI (Beginning and Inside) tags, which have proven successful in named-entity recognition. Since there are three labels in which we are interested (prefix, stem, and suffix), we apply the beginning and inside notion to each of them to create six tags: BEGINNING-PREFIX, INSIDE-PREFIX, BEGINNING-STEM, INSIDE-STEM, BEGINNING-SUFFIX, and INSIDE-SUFFIX. We train an MEMM to predict one of these six tags for each consonant. Furthermore, we constrain the decoder to allow only legal possible transitions given the current prediction, so that prefixes must come before stems and stems before suffixes. In order to capture the unknown word distributions, we train the consonant-level model on words occurring only once during training.

We call this word- and consonant-level segmentation model **hybrid**. As far as we are aware, this is a novel approach to segmentation.

2.3 Dictionary Linkage

For dictionary linkage, we divide the problem into two separate tasks: baseform linkage and root linkage. For both of these tasks, we use a hybrid model similar to that used for segmentation, consisting of a collection of separate MEMMs for each word type (either a stem or baseform, depending on the linker) and a model for unknown (or rare) words. For the unknown words, we compare two distinct approaches.

The first approach for unknown words is based on the work of Chrupała (2006), including the Morfette system. Instead of predicting a baseform given a stem, we predict what Chrupała calls a lemma-class. A lemma-class is the transformation specified by the minimum edit distance between the baseform (which he calls a lemma) and the stem. The transformation is a series of tuples, where each tuple includes (1) whether it was an insertion or deletion, (2) the letter inserted or deleted, and (3) the position of the insertion or deletion in the string (positions begin at zero). All operations are assumed to occur sequentially, as in Morfette. For example, the transformation of *XE;N* to *XEA* would proceed as follows: delete *;* from position 2, insert *A* into position 2, delete *N* from position 3.

In **hybrid-morfette** baseform linkage (respectively, root linkage), we predict a lemma-class (i.e., transformation) for each baseform (respectively, root). The predicted transformation is then applied to the stem (respectively, baseform) in order to construct the actual target baseform (respectively, root). The advantage to this method is that common transformations are grouped into a single class, thereby allowing the model to generalize and adequately predict baseforms (and roots) that have not been seen during training, but whose transformations have been seen. This model is trained on all words in order to capture as many transformations as possible.

The second approach for unknown words, called **hybrid-maxent**, uses an MEMM trained on all words seen in training. Given a stem (respectively, baseform), this approach predicts only baseforms (respectively, roots) that were observed in training data. Thus, this method has a distinct disadvantage when it comes to predicting new forms. This approach corresponds directly to the approach to handling unknown -words by Toutanova and Manning (2000) for POS tagging.

With regard to baseform and root linkage, we do not use the dictionary to constrain possible baseforms or roots, since we make no initial assumptions about the completeness of a dictionary.

2.4 Morphological Tagging

For morphological tagging, we break the task into two separate tasks: tagging the suffix and tagging the stem. Since there are a number of values that need to be predicted, we define two ways to approach the problem. We call the first approach the monolithic approach, in which the label is the concatenation of all the morphological attribute values. Table 3 illustrates the tagging of an example sentence: the stem tag and suffix tag columns contain the monolithic tags for stem tagging and suffix tagging. We use an MEMM to predict a monolithic tag for each stem or suffix and call this model **maxent-mono**. No co-occurrence restrictions among related or complementary morphological tags are directly enforced. Co-occurrence patterns are observed in the data, learned, and encoded in the models of the tagging process. It is worth noting further that constraints provided by the baseforms – predicted by dictionary linkage – on the morphological attributes are likewise not directly enforced. Enforcement of such constraints would require an infusion of expert knowledge into the system.

The second approach is to assume that morphological attributes are independent of each other. We call this the independent approach. Here, each tag is predicted by a tagger for a single morphological attribute. For example, the gender model is ignorant of the other 11 sub-tags during stem tagging. Using its local context (which does not include other stem sub-tags), the model predicts the best gender for a given word. The top prediction of each of these taggers (12, for stem tagging) is then combined naïvely with no notion of what combinations may be valid or invalid. We use MEMMs for each of the single-attribute taggers. This model is called **maxent-ind**.

2.5 Decoding

Our per-task decoders are beam decoders, with beam-size b. In particular, we limit the number of per-stage back-pointers to b due to the large size of the tagset for some of our sub-models. Although Viterbi decoding produces the most probable label sequence given a sequence of unlabeled words, it is potentially intractible on our hybrid models due to the unbounded dependence on previous consonant-level decisions. Our beam decoders produce a good approximation when tuned properly.

Decoding in **syromorph** consists of extending the per-task decoders to allow transitions from each sub-model to the next sub-model in the pipe. For example, in our pipeline, the first sub-model is segmentation. We predict the top n segmentations for the sentence (i.e., sequences of segmentations), where n is the number of transitions to maintain between each sub-task. Then, we run the remaining sub-tasks with each of the n sequences as a possible context. After each sub-task is completed, we narrow the number of possible contexts back to n.

We swept b and n for various values, and found $b = 5$ and $n = 5$ to be good values that balanced between accuracy and time; larger values saw only minute gains in accuracy.

Word	Transliteration	Pre.	Stem	Suffix	Baseform	Root	Suff. Tags	Stem Tags
ܘܥܒܕܬ	OEBDT	O	EBDT		EBD	EBD	0000	011012200000
ܐܢܘܢ	ANON		ANON		HO	HO	0000	300023222000
ܠܐܠܗܢ	LALHN	L	ALH	N	ALHA	ALH	1011	200310200200
ܡܠܟܘܬܐ	MLCOTA		MLCOTA		MLCOTA	MLC	0000	200310300200
ܘܟܗ̈ܢܐ	OCHNA	O	CHNA		CHNA	CHN	0000	200320200200
ܘܡܠ̈ܟܐ	OMLCA	O	MLCA		MLCA	MLC	0000	200320200200

Table 3: Part of a labeled Syriac sentence ܘܥܒܕܬ ܐܢܘܢ ܠܐܠܗܢ ܡܠܟܘܬܐ ܘܟܗ̈ܢܐ ܘܡܠ̈ܟܐ, "And you have made them a kingdom and priests and kings for our God." (Revelation 5:10)

3 Experimental Setup

We are using the Syriac Peshitta New Testament in the form compiled by Kiraz (1994).[2] This data is segmented, annotated with baseform and root, and labeled with morphological attributes. Kiraz and others in the Syriac community refined and corrected the original annotation while preparing a digital and print concordance of the New Testament. We augmented Kiraz's version of the data by segmenting suffixes and by streamlining the tagset. The dataset consists of 109,640 word tokens.

Table 3 shows part of a tagged Syriac sentence using this tagset. The suffix and stem tags consist of indices representing morphological attributes. In the example sentence, the suffix tag 1011 represents the values "masculine", "N/A", "plural", "normal suffix" for the suffix attributes of gender, person, number, and contraction. Each value of 0 for each stem and suffix attribute represents a value of "N/A", except for that of grammatical category, which always must have a value other than "N/A". Therefore, the suffix tag 0000 means there is no suffix.

For the stem tags, the attribute order is the same as that shown in Table 1 from top to bottom. The following describes the interpretation of the stem values represented in Table 3. Grammatical category values 0, 2, and 3 represent "verb", "noun", and "pronoun", respectively. (Grammatical category has no "N/A" value.) The verb conjugation value 1 represents "peal conjugation". Aspect value 1 represents "perfect". State value 3 represents "emphatic". Number values 1 and 2 represent "singular" and "plural". Person values 2 and 3 represent "second" and "third" person. Gender values 2 and 3 represent "masculine" and "feminine". Pronoun type value 2 represents "demonstrative". Demonstrative category value 2 represents "far". Finally, noun type 2 represents "common". The last two columns of 0 represent "N/A" for numeral type and particle type.

We implement five sub-tasks: segmentation, baseform linkage, root linkage, suffix tagging, and stem tagging. We compare each sub-task to a naïve approach as a baseline. In addition to desiring good sub-models, we also want a joint pipeline model that significantly outperforms the naïve joint approach, which is formed by using each of the following baselines in the pipeline framework.

The baseline implementation of segmentation is to choose the most-frequent label: for a given word, the baseline predicts the segmentation with which that word appeared most frequently during training. For unknown words, it chooses the largest prefix and largest suffix that is possible for that word from the list of prefixes and suffixes seen during training. (This naïve baseline for unknown words does not take into account the fact that the stem is often at least three characters in length.)

For dictionary linkage, the baseline is similar: both baseform linkage and root linkage use the most-frequent label approach. Given a stem, the baseline baseform linker predicts the baseform with which the stem was seen most frequently during training; likewise, the baseline root linker predicts the root from the baseform in a similar manner. For the unknown stem case, the baseline baseform linker predicts the baseform to be identical to the stem. For the unknown baseform case, the baseline root linker predicts a root identical to the first three consonants of the baseform, since for Syriac the root is exactly

[2] The Way International, a Biblical research ministry, annotated this version of the New Testament by hand and required 15 years to do so.

three consonants in a large majority of the cases.

The baselines for stem and suffix tagging are the most-frequent label approaches. These baselines are similar to **maxent-mono** and **maxent-ind**, using the monolithic and independent approaches used by **maxent-mono** and **maxent-ind**. The difference is that instead of using maximum entropy, the naïve most-frequent approach is used in its place.

The joint baseline tagger uses each of the component baselines in the n-best joint pipeline framework. Because this framework is modular, we can trivially swap in and out different models for each of the subtasks.

4 Experimental Results

Since we are focusing on under-resourced circumstances, we sweep the amount of training data and produce learning curves to better understand how our models perform in such circumstances. For each point in our learning curves and for all other evaluations, we employ ten-fold cross-validation. The learning curves use the chosen percentage of the data for training and a fixed-size test set from each fold and report the average accuracy.

The reported task accuracy requires the entire output for that task to be correct in order to be counted as correct. For example, during stem tagging, if one of the sub-tags is incorrect, then the entire tag is said to be incorrect. Furthermore, for **syromorph**, the outputs of every sub-task must be correct in order for the word token to be counted as correct.

Moving beyond token-level metrics, in order to understand performance of the system at the level of individual decisions (including N/A decisions), we compute decision-level accuracy: we call this metric **total-decisions**. For the **syromorph** method reported here, there are a total of 20 decisions: 2 for segmentation (prefix and suffix boundaries), 1 for baseform linkage, 1 for root linkage, 4 for suffix tagging, and 12 for stem tagging. This accuracy helps us to assess the number of decisions a human annotator would need to correct, if data were preannotated by a given model. Excluding N/A decisions, we compute per-decision coverage and accuracy. These metrics are called **applicable-coverage** and **applicable-accuracy**.

We show results on both the individual sub-tasks and the entire joint task. Since previous subtasks can adversely affect tasks further down in the pipeline, we evaluate the sub-models by placing them in the pipeline with other (simulated) submodels that correctly predict every instance. For example, when testing a root linker, we place the root linker to be evaluated in the pipeline with a segmenter, baseform linker, and taggers that return the correct label for every prediction. This gives an upper-bound for the individual model, removes the possibility of error propagation, and shows how well that model performs without the effects of the other models in the pipeline.

For our results, unknown accuracy is the accuracy of unknown instances, specific to the task, at training time. In the case of baseform linkage, for example, a stem is considered unknown if that stem was not seen during training. It is therefore possible to have a known word with an unknown stem and vice versa. As in other NLP problems, unknown instances are a manifestation of training data sparsity.

4.1 Baseline Results

Table 4 is grouped by sub-task and reports the results of each of the baseline sub-tasks in the first row of each group. Each of the baselines performs surprisingly well. The accuracies of the baselines for most of the tasks are high because the ambiguity of the labels given the instance is quite low: the average ambiguity across word types for segmentation, baseform linkage, root linkage, suffix tagging, and stem tagging are 1.01, 1.05, 1.02, 1.35, and 1.47, respectively.

Preliminary experiments indicated that if we had trained a baseline model using a single prediction (a monolithic concatenation of the predictions for all tasks) per token rather than separating the tasks, the baseline tagging accuracy would have been lower. Note that the unknown tagging accuracy for the monolithic suffix tagger is not applicable, because there were no test suffixes that were not seen during training.

4.2 Individual Model Results

Table 4 also shows the results for the individual models. In the table, SEG, BFL, RTL, SUFFIX, and STEM represent segmentation, baseform linkage, root linkage, suffix tagging, and stem tagging,

	Model	Total	Known	Unk
SEG	baseline	96.75	99.64	69.11
	hybrid	**98.87**	**99.70**	**90.83**
BFL	baseline	95.64	98.45	22.28
	hybrid-morfette	**96.19**	98.05	**78.40**
	hybrid-maxent	**96.19**	**99.15**	67.86
RTL	baseline	98.84	**99.56**	80.20
	hybrid-morfette	**99.05**	99.44	**88.86**
	hybrid-maxent	98.34	99.45	69.30
SUFFIX	mono. baseline	98.75	98.75	N/A
	ind. baseline	96.74	98.78	**0.01**
	maxent-mono	**98.90**	**98.90**	N/A
	maxent-ind	**98.90**	**98.90**	N/A
STEM	mono. baseline	83.08	86.26	0.01
	ind. baseline	53.24	86.90	0.00
	maxent-mono	**89.48**	**92.87**	**57.04**
	maxent-ind	88.43	90.26	40.59

Table 4: Word-level accuracies for the individual submodels used in the **syromorph** approach.

respectively. Even though the baselines were high, each individual model outperformed its respective baseline, with the exception of the root linker. Two of the most interesting results are the known accuracy of the baseform linkers **hybrid-maxent** and **hybrid-morfette**. As hybrid models, the difference between them lies only in the treatment of unknown words; however, the known accuracy of the morfette model drops fairly significantly. This is due to the unknown words altering the weights for features in which those words occur. For instance, if the previous word is unknown and a baseform that was never seen was predicted, then the weights on the next word for all features that contain that unknown word will be quite different than if that previous word were a known word.

It is also worth noting that the stem tagger is by far the worst model in this group of models, but it is also the most difficult task. The largest gains in improving the entire system would come from focusing attention on that task.

4.3 Joint Model Results

Table 5 shows the accuracies for the joint models. The joint model incorporating "maxent" variants performs best overall and on known cases. The joint model incorporating the "morfette" variants performs best on unknown cases.

Model	Total	Known	Unk
Baseline	80.76	85.74	28.07
Morfette Monolithic	85.96	89.85	**44.86**
Maxent Monolithic	**86.47**	**90.77**	40.93

Table 5: Word-level accuracies for various joint **syromorph** models.

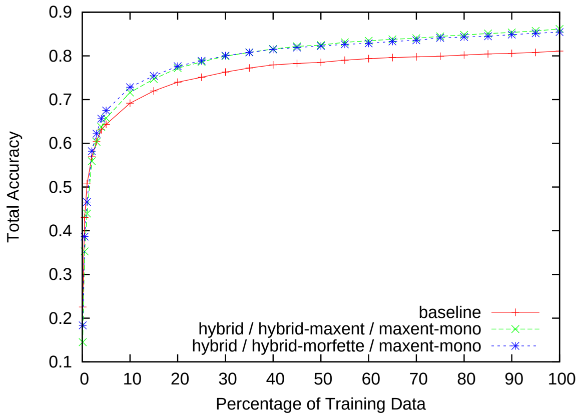


Figure 2: The total accuracy of the joint model.

Decision-level metrics for the SEG:**hybrid** / BFL and RTL:**hybrid-maxent** / SUFFIX and STEM:**maxent-mono** model are as follows: for **total-decisions**, the model achieves an accuracy of 97.08%, compared to 95.50% accuracy for the baseline, amounting to a 35.11% reduction in error rate over the baseline; for **applicable-coverage** and **applicable-accuracy** this model achieved 93.45% and 93.81%, respectively, compared to the baseline's 90.03% and 91.44%.

Figures 2, 3, and 4 show learning curves for total, known, and unknown accuracies for the joint pipeline model. As can be seen in Figure 2, by the time we reach 10% of the training data, **syromorph** is significantly better than the baseline. In fact, at 35% of the training data, our joint pipeline model outperforms the baseline trained with all available training data.

Figure 3 shows the baseline performing quite well on known words with very low amounts of data. Since the x-axis varies the amount of training data, the meaning of "known" and "unknown" evolves as we move to the right of the graph; consequently, the

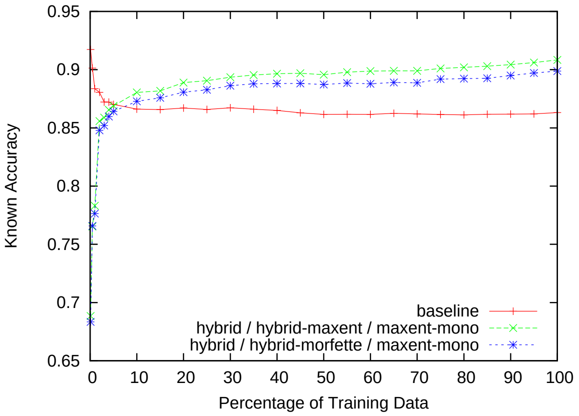


Figure 3: The accuracy of the joint model on known words.

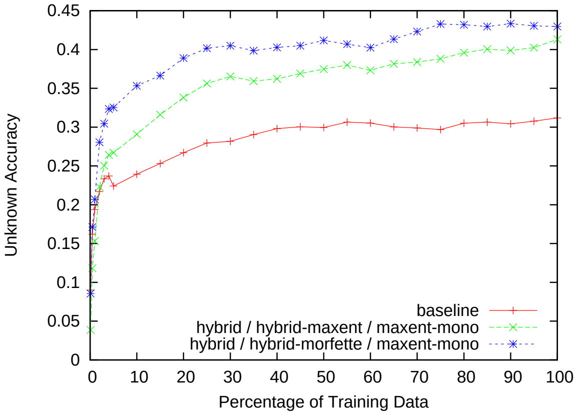


Figure 4: The accuracy of the joint model on unknown words.

left and right sides of the graph are incomparable. When the percentage of training data is very low, the percentage of unknown words is high, and the number of known words is relatively low. On this dataset, the more frequent words tend to be less ambiguous, giving the most-frequent taggers an advantage in a small random sample. For this reason, the baseline performs very well on known accuracy with lower amounts of training data.

Figure 4 clearly shows that **hybrid-morfette** linkers outperform **hybrid-maxent** linkers on unknown words. However, Figures 2- 4 show that **hybrid-morfette**'s advantage on unknown words is counteracted by its lower performance on known words; therefore, it has slightly lower overall accuracy than **hybrid-maxent**.

5 Related Work

The most closely related work to our approach is the Morfette tool for labeling inflectional morphology (Chrupała et al., 2008). Chrupała et al. created a tool that labels Polish, Romanian, and Spanish with morphological information as well as baseforms. It is a supervised learning approach that requires data labeled with both morphological tags and baseforms. This approach creates two separate models (a morphological tagger and a lemmatizer) and combines the decoding process in order to create a joint model that predicts both morphological tags and the baseform. Morfette uses MEMMs for both models and has access to predicted labels in the feature set. Reported accuracy rates are 96.08%, 93.83%, and 81.19% for joint accuracy on datasets trained with fewer than 100,000 tokens for Romanian, Spanish, and Polish, respectively. The major difference between this work and ours is the degree of morphological analysis required by the languages. Chrupała et al. neglect segmentation, a task not as intuitive for their languages as it is for Syriac. These languages also require only linkage to a baseform, as no root exists.

Also closely related is the work of Daya, Roth, and Wintner (2008) on Hebrew. The authors use the notion of patterns into which root consonants are injected to compose Semitic words. They employ linguistic knowledge (specifically, lists of prefixes, suffixes, and "knowledge of word-formation processes" combined with SNoW, a multi-class classifier that has been shown to work well in other NLP tasks. The major difference between this approach and the method presented in this paper is that this method does not require the extra knowledge required to encode word-formation processes. A further point of difference is our use of hybrid word- and consonant-level models, after Haertel et al. (2010). Their work builds on the work of Shacham and Wintner (2007), which is also related to that of Habash and Rambow, described below.

Work by Lee et al. (2003) is the most relevant work for segmentation, since they segment Arabic, closely related to Syriac, with a data-driven approach. Lee et al. use an unsupervised algorithm bootstrapped with manually segmented data to learn the segmentation for Arabic without any additional language resources. At the heart of the algorithm is a word-level trigram language model, which captures the correct weights for prefixes and suffixes. They report an accuracy of 97%. We opted to use our own segmenter because we felt we could achieve higher accuracy with the **hybrid** segmenter.

Mohamed and Kübler (2010a, 2010b) report on closely related work for morphological tagging. They use a data-driven approach to find the POS tags for Arabic, using both word tokens and segmented words as inputs for their system. Although their segmentation performance is high, they report that accuracy is lower when first segmenting word tokens. They employ TiMBL, a memory-based learner, as their model and report an accuracy of 94.74%.

Habash and Rambow (2005) currently have the most accurate approach for Arabic morphological analysis using additional language tools. They focus on morphological disambiguation (tagging), given morphological segmentation in the output of the morphological analyzer. For each word, they first run it through the morphological analyzer to reduce the number of possible outputs. They then train a separate Support Vector Machine (SVM) for each morphological attribute (ten in all). They look at different ways of combining these outputs to match an output from the morphological analyzer. For their best model, they report an overall tag accuracy of 97.6%.

Others have used morphological analyzers and other language tools for morphological disambiguation coupled with segmentation. The following works exemplify this approach: Diab et al. (2004) use a POS tagger to jointly segment, POS tag, and chunk base-phrases for Arabic with SVMs. Kudo et al. (2004) use SVMs to morphologically tag Japanese. Smith et al. (2005) use SVMs for segmentation, lemmatization, and POS tagging for Arabic, Korean, and Czech. Petkevič (2001) use a morphological analyzer and additional simple rules for morphological disambiguation of Czech. Mansour et al. (2007) and Bar-haim et al. (2008) both use hidden Markov models to POS tag Hebrew, with the latter including segmentation as part of the task.

For Syriac, a morphological analyzer is not available. Kiraz (2000) created a Syriac morphological analyzer using finite-state methods; however, it was developed on outdated and now inaccessible equipment and is no longer working or available to us.

6 Conclusions and Future Work

We have shown that we can effectively model segmentation, linkage to headwords in a dictionary, and morphological tagging using a joint model called **syromorph**. We have introduced novel approaches for segmentation, dictionary linkage, and morphological tagging, and each of these approaches has outperformed its corresponding naïve baseline. Furthermore, we have shown that for Syriac, a data-driven approach seems to be an appropriate way to solve these problems in an under-resourced setting.

We hope to use this combined model for pre-annotation in an active learning setting to aid annotators in labeling a large Syriac corpus. This corpus will contain data spanning multiple centuries and a variety of authors and genres. Future work will require addressing issues encountered in this corpus. In addition, there is much to do in getting the overall tag accuracy closer to the accuracy of individual decisions. We leave further feature engineering for the stem tagger and the exploration of possible new morphological tagging techniques for future work.

Finally, future work includes the application of the **syromorph** methodology to other under-resourced Semitic languages.

Acknowledgments

We would like to thank David Taylor of the Oriental Institute at Oxford University for collaboration on the design of the simplified tagset. We also recognize the assistance of Ben Hansen of BYU on a subset of the experimental results. Finally, we would like to thank the anonymous reviewers for helpful guidance.

References

Roy Bar-haim, Khalil Sima'an, and Yoad Winter. 2008. Part-of-speech tagging of modern hebrew text. *Natural Language Engineering*, 14(2):223–251.

British and Foreign Bible Society, editors. 1920. *The New Testament in Syriac*. Oxford: Frederick Hall.

Grzegorz Chrupała, Georgiana Dinu, and Josef van Genabith. 2008. Learning morphology with Morfette. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*.

Grzegorz Chrupała. 2006. Simple data-driven context-sensitive lemmatization. In *Procesamiento del Lenguaje Natural*, volume 37, pages 121 – 127.

Ezra Daya, Dan Roth, and Shuly Wintner. 2008. Identifying Semitic roots: Machine learning with linguistic constraints. *Computational Linguistics*, 34(3):429–448.

Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. 2004. Automatic tagging of Arabic text: From raw text to base phrase chunks. In *Proceedings of the 5th Meeting of the North American Chapter of the Association for Computational Linguistics/Human Language Technologies Conference (HLT-NAACL04)*, pages 149–152.

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 *EMNLP '06: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 618–626. Association for Computational Linguistics.

Nizar Habash and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 573–580. Association for Computational Linguistics.

Nizar Habash and Owen Rambow. 2007. Arabic diacritization through full morphological tagging. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 53–56. Association for Computational Linguistics.

Robbie Haertel, Peter McClanahan, and Eric K. Ringger. 2010. Automatic diacritization for low-resource languages using a hybrid word and consonant cmm. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 519–527. Association for Computational Linguistics.

Jan Hajič and Barbora Hladká. 1998. Tagging inflective languages: Prediction of morphological categories for a rich, structured tagset. In *Proceedings of the 17th International Conference on Computational Linguistics*, pages 483–490. Association for Computational Linguistics.

George Kiraz. 1994. Automatic concordance generation of Syriac texts. In R. Lavenant, editor, *VI Symposium Syriacum 1992*, pages 461–.

George Anton Kiraz. 2000. Multitiered nonlinear morphology using multitape finite automata: a case study on Syriac and Arabic. *Computational Linguistics*, 26:77–105.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to japanese morphological analysis. In *Proceedings of EMNLP*, pages 230–237.

Young-Suk Lee, Kishore Papineni, Salim Roukos, Ossama Emam, and Hany Hassan. 2003. Language model based Arabic word segmentation. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 399–406. Association for Computational Linguistics.

Saib Mansour, Khalil Sima'an, and Yoad Winter. 2007. Smoothing a lexicon-based pos tagger for Arabic and Hebrew. In *Semitic '07: Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages*, pages 97–103. Association for Computational Linguistics.

Emad Mohamed and Sandra Kübler. 2010a. Arabic part of speech tagging. In *Proceedings of the Seventh International Language Resources and Evaluation (LREC'10)*.

Emad Mohamed and Sandra Kübler. 2010b. Is Arabic part of speech tagging feasible without word segmentation? In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 705–708. Association for Computational Linguistics.

Vladimír Petkevič. 2001. Grammatical agreement and automatic morphological disambiguation of inflectional languages. In *TSD '01: Proceedings of the 4th International Conference on Text, Speech and Dialogue*, pages 47–53. Springer-Verlag.

Danny Shacham and Shuly Wintner. 2007. Morphological disambiguation of Hebrew: a case study in classifier combination. In *Proceedings of EMNLP-CoNLL 2007, the Conference on Empirical Methods in Natural Language Processing and the Conference on Computational Natural Language Learning*. Association for Computational Linguistics.

Noah A. Smith, David A. Smith, and Roy W. Tromble. 2005. Context-based morphological disambiguation with random fields. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 475–482. Association for Computational Linguistics.

K. Toutanova and C. Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of EMNLP*, pages 63–70.

Lessons Learned in Part-of-Speech Tagging of Conversational Speech

Vladimir Eidelman[†], **Zhongqiang Huang**[†], **and Mary Harper**[†‡]
[†]Laboratory for Computational Linguistics and Information Processing
Institute for Advanced Computer Studies
University of Maryland, College Park, MD
[‡]Human Language Technology Center of Excellence
Johns Hopkins University, Baltimore, MD
{vlad,zhuang,mharper}@umiacs.umd.edu

Abstract

This paper examines tagging models for spontaneous English speech transcripts. We analyze the performance of state-of-the-art tagging models, either generative or discriminative, left-to-right or bidirectional, with or without latent annotations, together with the use of ToBI break indexes and several methods for segmenting the speech transcripts (i.e., conversation side, speaker turn, or human-annotated sentence). Based on these studies, we observe that: (1) bidirectional models tend to achieve better accuracy levels than left-to-right models, (2) generative models seem to perform somewhat better than discriminative models on this task, and (3) prosody improves tagging performance of models on conversation sides, but has much less impact on smaller segments. We conclude that, although the use of break indexes can indeed significantly improve performance over baseline models without them on conversation sides, tagging accuracy improves more by using smaller segments, for which the impact of the break indexes is marginal.

1 Introduction

Natural language processing technologies, such as parsing and tagging, often require reconfiguration when they are applied to challenging domains that differ significantly from newswire, e.g., blogs, twitter text (Foster, 2010), or speech. In contrast to text, conversational speech represents a significant challenge because the transcripts are not segmented into sentences. Furthermore, the transcripts are often disfluent and lack punctuation and case information. On the other hand, speech provides additional information, beyond simply the sequence of words, which could be exploited to more accurately assign each word in the transcript a part-of-speech (POS) tag. One potentially beneficial type of information is prosody (Cutler et al., 1997).

Prosody provides cues for lexical disambiguation, sentence segmentation and classification, phrase structure and attachment, discourse structure, speaker affect, etc. Prosody has been found to play an important role in speech synthesis systems (Batliner et al., 2001; Taylor and Black, 1998), as well as in speech recognition (Gallwitz et al., 2002; Hasegawa-Johnson et al., 2005; Ostendorf et al., 2003). Additionally, prosodic features such as pause length, duration of words and phones, pitch contours, energy contours, and their normalized values have been used for speech processing tasks like sentence boundary detection (Liu et al., 2005).

Linguistic encoding schemes like ToBI (Silverman et al., 1992) have also been used for sentence boundary detection (Roark et al., 2006; Harper et al., 2005), as well as for parsing (Dreyer and Shafran, 2007; Gregory et al., 2004; Kahn et al., 2005). In the ToBI scheme, aspects of prosody such as tone, prominence, and degree of juncture between words are represented symbolically. For instance, Dreyer and Shafran (2007) use three classes of automatically detected ToBI break indexes, indicating major intonational breaks with a `4`, hesitation with a `p`, and all other breaks with a `1`.

Recently, Huang and Harper (2010) found that they could effectively integrate prosodic informa-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 821–831,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

tion in the form of this simplified three class ToBI encoding when parsing spontaneous speech by using a prosodically enriched PCFG model with latent annotations (PCFG-LA) (Matsuzaki et al., 2005; Petrov and Klein, 2007) to rescore n-best parses produced by a baseline PCFG-LA model without prosodic enrichment. However, the prosodically enriched models by themselves did not perform significantly better than the baseline PCFG-LA model without enrichment, due to the negative effect that misalignments between automatic prosodic breaks and true phrase boundaries have on the model.

This paper investigates methods for using state-of-the-art taggers on conversational speech transcriptions and the effect that prosody has on tagging accuracy. Improving POS tagging performance of speech transcriptions has implications for improving downstream applications that rely on accurate POS tags, including sentence boundary detection (Liu et al., 2005), automatic punctuation (Hillard et al., 2006), information extraction from speech, parsing, and syntactic language modeling (Heeman, 1999; Filimonov and Harper, 2009). While there have been several attempts to integrate prosodic information to improve parse accuracy of speech transcripts, to the best of our knowledge there has been little work on using this type of information for POS tagging. Furthermore, most of the parsing work has involved generative models and rescoring/reranking of hypotheses from the generative models. In this work, we will analyze several factors related to effective POS tagging of conversational speech:

- discriminative versus generative POS tagging models (Section 2)
- prosodic features in the form of simplified ToBI break indexes (Section 4)
- type of speech segmentation (Section 5)

2 Models

In order to fully evaluate the difficulties inherent in tagging conversational speech, as well as the possible benefits of prosodic information, we conducted experiments with six different POS tagging models. The models can be broadly separated into two classes: generative and discriminative. As the first of our generative models, we used a Hidden Markov Model (HMM) trigram tagger (Thede and Harper, 1999), which serves to establish a baseline and to gauge the difficulty of the task at hand. Our second model, HMM-LA, was the latent variable bigram HMM tagger of Huang et al. (2009), which achieved state-of-the-art tagging performance by introducing latent tags to weaken the stringent Markov independence assumptions that generally hinder tagging performance in generative models.

For the third model, we implemented a bidirectional variant of the HMM-LA (HMM-LA-Bidir) that combines evidence from two HMM-LA taggers, one trained left-to-right and the other right-to-left. For decoding, we use a product model (Petrov, 2010). The intuition is that the context information from the left and the right of the current position is complementary for predicting the current tag and thus, the combination should serve to improve performance over the HMM-LA tagger.

Since prior work on parsing speech with prosody has relied on generative models, it was necessary to modify equations of the model in order to incorporate the prosodic information, and then perform rescoring in order to achieve gains. However, it is far simpler to directly integrate prosody as features into the model by using a discriminative approach. Hence, we also investigate several log-linear models, which allow us to easily include an arbitrary number and varying kinds of possibly overlapping and non-independent features.

First, we implemented a Conditional Random Field (CRF) tagger, which is an attractive choice due to its ability to learn the globally optimal labeling for a sequence and proven excellent performance on sequence labeling tasks (Lafferty et al., 2001). In contrast to an HMM which optimizes the joint likelihood of the word sequence and tags, a CRF optimizes the conditional likelihood, given by:

$$p_\lambda(t|w) = \frac{\exp \sum_j \lambda_j F_j(t,w)}{\sum_t \exp \sum_j \lambda_j F_j(t,w)} \qquad (1)$$

where the λ's are the parameters of the model to estimate and F indicates the feature functions used. The denominator in (1) is $Z_\lambda(x)$, the normalization factor, with:

$$F_j(t,w) = \sum_i f_j(t,w,i)$$

Class	Model Name	Latent Variable	Bidirectional	N-best-Extraction	Markov Order
Generative	Trigram HMM			√	2nd
	HMM-LA	√		√	1st
	HMM-LA-Bidir	√	√		1st
Discriminative	Stanford Bidir		√		2nd
	Stanford Left5				2nd
	CRF				2nd

Table 1: Description of tagging models

The objective we need to maximize then becomes :

$$\mathcal{L} = \sum_n \left[\sum_j \lambda_j F_j(t_n, w_n) - \log Z_\lambda(x_n) \right] - \frac{\|\lambda\|^2}{2\sigma^2}$$

where we use a spherical Gaussian prior to prevent overfitting of the model (Chen and Rosenfeld, 1999) and the wide-spread quasi-Newtonian L-BFGS method to optimize the model parameters (Liu and Nocedal, 1989). Decoding is performed with the Viterbi algorithm.

We also evaluate state-of-the-art Maximum Entropy taggers: the Stanford Left5 tagger (Toutanova and Manning, 2000) and the Stanford bidirectional tagger (Toutanova et al., 2003), with the former using only left context and the latter bidirectional dependencies.

Table 1 summarizes the major differences between the models along several dimensions: (1) generative versus discriminative, (2) directionality of decoding, (3) the presence or absence of latent annotations, (4) the availability of n-best extraction, and (5) the model order.

In order to assess the quality of our models, we evaluate them on the section 23 test set of the standard newswire WSJ tagging task after training all models on sections 0-22. Results appear in Table 2. Clearly, all the models have high accuracy on newswire data, but the Stanford bidirectional tagger significantly outperforms the other models with the exception of the HMM-LA-Bidir model on this task.[1]

Model	Accuracy
Trigram HMM	96.58
HMM-LA	97.05
HMM-LA-Bidir	97.16
Stanford Bidir	97.28
Stanford Left5	97.07
CRF	96.81

Table 2: Tagging accuracy on WSJ

3 Experimental Setup

In the rest of this paper, we evaluate the tagging models described in Section 2 on conversational speech. We chose to utilize the Penn Switchboard (Godfrey et al., 1992) and Fisher treebanks (Harper et al., 2005; Bies et al., 2006) because they provide gold standard tags for conversational speech and we have access to corresponding automatically generated ToBI break indexes provided by (Dreyer and Shafran, 2007; Harper et al., 2005)[2].

We utilized the Fisher dev1 and dev2 sets containing 16,519 sentences (112,717 words) as the primary training data and the entire Penn Switchboard treebank containing 110,504 sentences (837,863 words) as an additional training source[3]. The treebanks were preprocessed as follows: the tags of auxiliary verbs were replaced with the AUX tag, empty nodes

[1] Statistically significant improvements are calculated using the sign test ($p < 0.05$).

[2] A small fraction of words in the Switchboard treebank do not align with the break indexes because they were produced based on a later refinement of the transcripts used to produce the treebank. For these cases, we heuristically added break *1* to words in the middle of a sentence and *4* to words that end a sentence.

[3] Preliminary experiments evaluating the effect of training data size on performance indicated using the additional Switchboard data leads to more accurate models, and so we use the combined training set.

and function tags were removed, words were downcased, punctuation was deleted, and the words and their tags were extracted. Because the Fisher treebank was developed using the lessons learned when developing Switchboard, we chose to use its eval portion for development (the first 1,020 tagged sentences containing 7,184 words) and evaluation (the remaining 3,917 sentences with 29,173 words).

We utilize the development set differently for the generative and discriminative models. Since the EM algorithm used for estimating the parameters in the latent variable models introduces a lot of variability, we train five models with a different seed and then choose the best one based on dev set performance. For the discriminative models, we tuned their respective regularization parameters on the dev set. All results reported in the rest of this paper are on the test set.

4 Integration of Prosodic Information

In this work, we use three classes of automatically generated ToBI break indexes to represent prosodic information (Kahn et al., 2005; Dreyer and Shafran, 2007; Huang and Harper, 2010): `4`, `1`, and `p`. Consider the following speech transcription example, which is enriched with ToBI break indexes in parentheses and tags: `i(1)/PRP did(1)/VBD n't(1)/RB you(1)/PRP know(4)/VBP i(1)/PRP did(1)/AUX n't(1)/RB...` The speaker begins an utterance, and then restarts the utterance. The automatically predicted break `4` associated with `know` in the utterance compellingly indicates an intonational phrase boundary and could provide useful information for tagging if we can model it appropriately.

To integrate prosody into our generative models, we utilize the method from (Dreyer and Shafran, 2007) to add prosodic breaks. As Figure 1 shows, ToBI breaks provide a secondary sequence of observations that is parallel to the sequence of words that comprise the sentence. Each break b_i in the secondary sequence is generated by the same tag t_i as that which generates the corresponding word w_i, and so it is conditionally independent of its corresponding word given the tag:

$$P(w, b|t) = P(w|t)P(b|t)$$

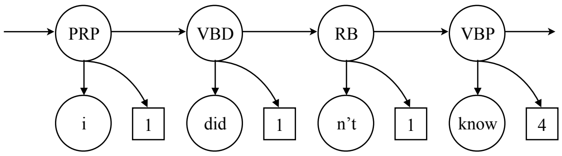


Figure 1: Parallel generation of words and breaks for the HMM models

The HMM-LA taggers are then able to split tags to capture implicit higher order interactions among the sequence of tags, words, and breaks.

The discriminative models are able to utilize prosodic features directly, enabling the use of contextual interactions with other features to further improve tagging accuracy. Specifically, in addition to the standard set of features used in the tagging literature, we use the feature templates presented in Table 3, where each feature associates the break b_i, word w_i, or some combination of the two with the current tag t_i[4].

Break and/or word values	Tag value
b_i=B	t_i = T
b_i=B & b_{i-1}=C	t_i = T
w_i=W & b_i=B	t_i = T
w_{i+1}=W & b_i=B	t_i = T
w_{i+2}=W & b_i=B	t_i = T
w_{i-1}=W & b_i=B	t_i = T
w_{i-2}=W & b_i=B	t_i = T
w_i=W & b_i=B & b_{i-1}=C	t_i = T

Table 3: Prosodic feature templates

5 Experiments

5.1 Conversation side segmentation

When working with raw speech transcripts, we initially have a long stream of unpunctuated words, which is called a conversation side. As the average length of conversation side segments in our data is approximately 630 words, it poses quite a challenging tagging task. Thus, we hypothesize that it is on these large segments that we should achieve the most

[4] We modified the Stanford taggers to handle these prosodic features.

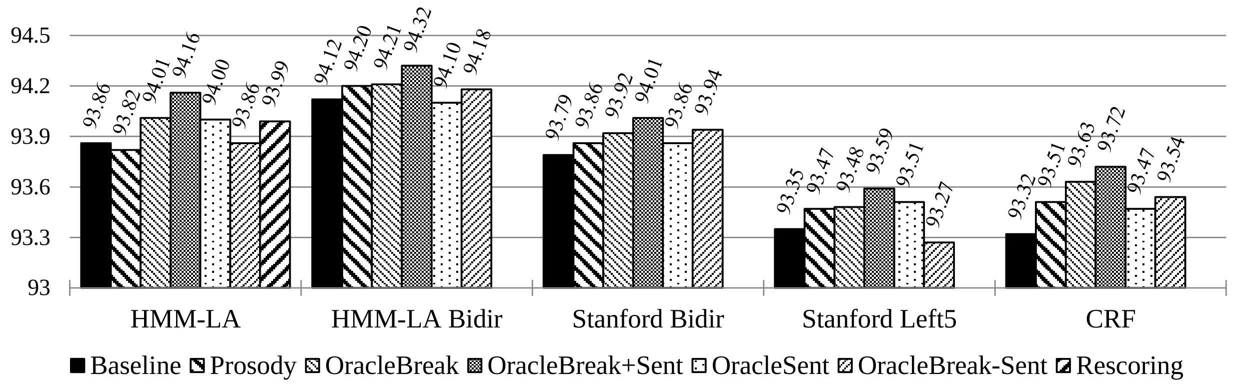


Figure 2: Tagging accuracy on conversation sides

improvement from the addition of prosodic information.

In fact, as the baseline results in Figure 2 show, the accuracies achieved on this task are much lower than those on the newswire task. The trigram HMM tagger accuracy drops to 92.43%, while all the other models fall to within the range of 93.3%-94.12%, a significant departure from the 96-97.3% range on newswire sentences. Note that the Stanford bidirectional and HMM-LA tagger perform very similarly, although the HMM-LA-Bidir tagger performs significantly better than both. In contrast to the newswire task on which the Stanford bidirectional tagger performed the best, on this genre, it is slightly worse than the HMM-LA tagger, albeit the difference is not statistically significant.

With the direct integration of prosody into the generative models (see Figure 2), there is a slight but statistically insignificant shift in performance. However, integrating prosody directly into the discriminative models leads to significant improvements in the CRF and Stanford Left5 taggers. The gain in the Stanford bidirectional tagger is not statistically significant, however, which suggests that the left-to-right models benefit more from the addition of prosody than bidirectional models.

5.2 Human-annotated sentences

Given the lack-luster performance of the tagging models on conversation side segments, even with the direct addition of prosody, we chose to determine the performance levels that could be achieved on this task using human-annotated sentences, which we will refer to as sentence segmentation. Figure 3 reports the baseline tagging accuracy on sentence segments, and we see significant improvements across all models. The HMM Trigram tagger performance increases to 93.00%, while the increase in accuracy for the other models ranges from around 0.2-0.3%. The HMM-LA taggers once again achieve the best performance, with the Stanford bidirectional close behind. Although the addition of prosody has very little impact on either the generative or discriminative models when applied to sentences, the baseline tagging models (i.e., not prosodically enriched) significantly outperform all of the prosodically enriched models operating on conversation sides.

At this point, it would be apt to suggest using automatic sentence boundary detection to create shorter segments. Table 4 presents the results of using baseline models without prosodic enrichment trained on the human-annotated sentences to tag automatically segmented speech[5]. As can be seen, the results are quite similar to the conversation side segmentation performances, and thus significantly lower than when tagging human-annotated sentences. A caveat to consider here is that we break the standard assumption that the training and test set be drawn from the same distribution, since the training data is human-annotated and the test is automatically segmented. However, it can be quite challenging to create a corpus to train on that represents the biases of the systems that perform automatic sentence segmentation. Instead, we will examine an-

[5] We used the Baseline Structural Metadata System described in Harper et al. (2005) to predict sentence boundaries.

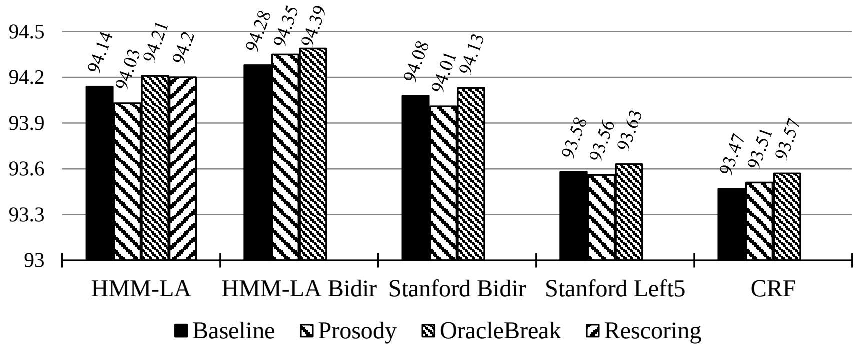


Figure 3: Tagging accuracy on human-annotated segments

other segmentation method to shorten the segments automatically, i.e., by training and testing on speaker turns, which preserves the train-test match, in Section 5.5.

Model	Accuracy
HMM-LA	93.95
HMM-LA-Bidir	94.07
Stanford Bidir	93.77
Stanford Left5	93.35
CRF	93.29

Table 4: Baseline tagging accuracy on automatically detected sentence boundaries

5.3 Oracle Break Insertion

As we believe one of the major roles that prosodic cues serve for tagging conversation sides is as a proxy for sentence boundaries, perhaps the efficacy of the prosodic breaks can, at least partially, be attributed to errors in the automatically induced break indexes themselves, as they can misalign with syntactic phrase boundaries, as discussed in Huang and Harper (2010). This may degrade the performance of our models more than the improvement achieved from correctly placed breaks. Hence, we conduct a series of experiments in which we systematically eliminate noisy phrase and disfluency breaks and show that under these improved conditions, prosodically enriched models can indeed be more effective.

To investigate to what extent noisy breaks are impeding the possible improvements from prosodically enriched models, we replaced all `4` and `p` breaks in the training and evaluation sets that did not align to the correct phrase boundaries as indicated by the treebank with break `1` for both the conversation sides and human-annotated sentences. The results from using Oracle Breaks on conversation sides can be seen in Figure 2. All models except Stanford Left5 and HMM-LA-Bidir significantly improve in accuracy when trained and tested on the Oracle Break modified data. On human-annotated sentences, Figure 3 shows improvements in accuracies across all models, however, they are statistically insignificant.

To further analyze why prosodically enriched models achieve more improvement on conversation sides than on sentences, we conducted three more Oracle experiments on conversation sides. For the first, OracleBreak-Sent, we further modified the data such that all breaks corresponding to a sentence ending in the human-annotated segments were converted to break `1`, thus effectively only leaving inside sentence phrasal boundaries. This modification results in a significant drop in performance, as can be seen in Figure 2.

For the second, OracleSent, we converted all the breaks corresponding to a sentence end in the human-annotated segmentations to break `4`, and all the others to break `1`, thus effectively only leaving sentence boundary breaks. This performed largely on par with OracleBreak, suggesting that the phrase-aligned prosodic breaks seem to be a stand-in for sentence boundaries.

Finally, in the last condition, OracleBreak+Sent, we modified the OracleBreak data such that all breaks corresponding to a sentence ending in the human-annotated sentences were converted to break

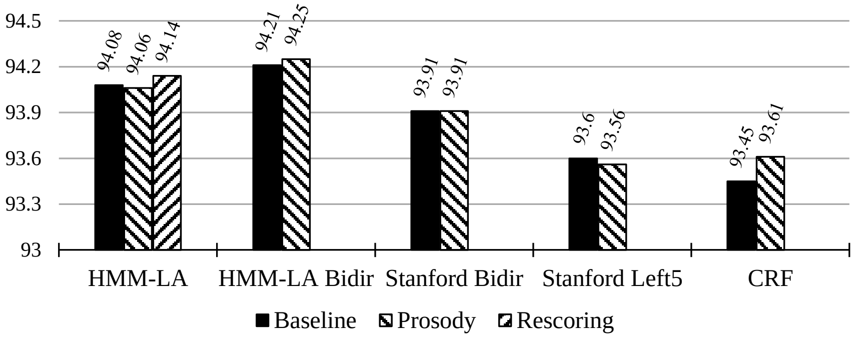


Figure 4: Tagging accuracy on speaker turns

4 (essentially combining OracleBreak and OracleSent). As Figure 2 indicates, this modification results in the best tagging accuracies for all the models. All models were able to match or even improve upon the baseline accuracies achieved on the human segmented data. This suggests that when we have breaks that align with phrasal and sentence boundaries, prosodically enriched models are highly effective.

5.4 N-best Rescoring

Based on the findings in the previous section and the findings of (Huang and Harper, 2010), we next apply a rescoring strategy in which the search space of the prosodically enriched generative models is restricted to the n-best list generated from the baseline model (without prosodic enrichment). In this manner, the prosodically enriched model can avoid poor tag sequences produced due to the misaligned break indexes. As Figure 2 shows, using the baseline conversation side model to produce an n-best list for the prosodically enriched model to rescore results in significant improvements in performance for the HMM-LA model, similar to the parsing results of (Huang and Harper, 2010). The size of the n-best list directly impacts performance, as reducing to $n = 1$ is akin to tagging with the baseline model, and increasing $n \to \infty$ amounts to tagging with the prosodically enriched model. We experimented with a number of different sizes for n and chose the best one using the dev set. Figure 3 presents the results for this method applied to human-annotated sentences, where it produces only marginal improvements[6].

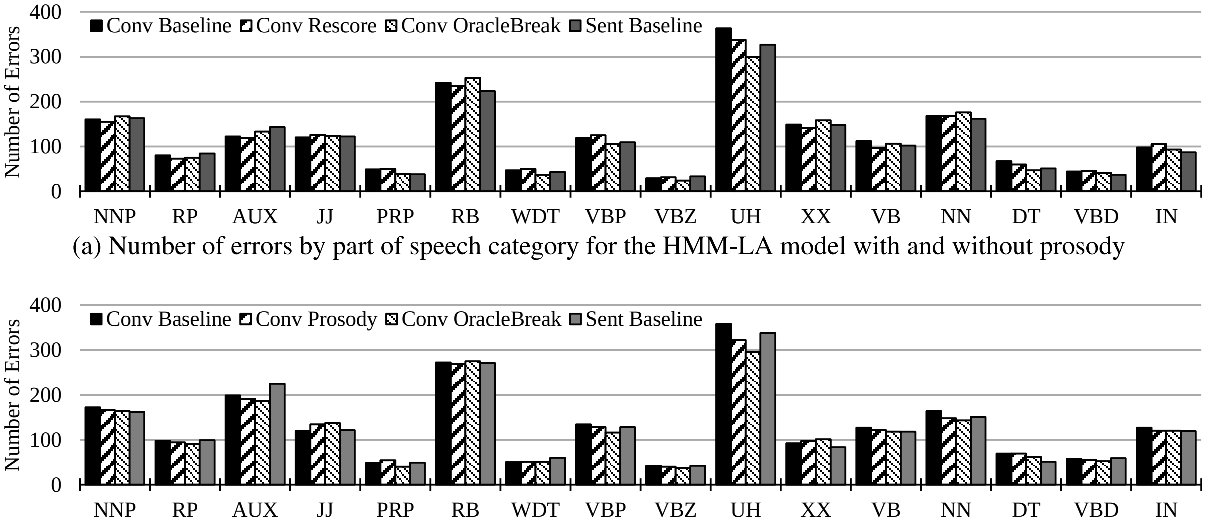
5.5 Speaker turn segmentation

The results presented thus far indicate that if we have access to close to perfect break indexes, we can use them effectively, but this is not likely to be true in practice. We have also observed that tagging accuracy on shorter conversation sides is greater than longer conversation sides, suggesting that post-processing the conversation sides to produce shorter segments would be desirable.

We thus devised a scheme by which we could automatically extract shorter speaker turn segments from conversation sides. For this study, speaker turns, which effectively indicate speaker alternations, were obtained by using the metadata in the treebank to split the sentences into chunks based on speaker change. Every time a speaker begins talking after the other speaker was talking, we start a new segment for that speaker. In practice, this would need to be done based on audio cues and automatic transcriptions, so these results represent an upper bound.

Figure 4 presents tagging results on speaker turn segments. For most models, the difference in accuracy achieved on these segments and that of human-annotated sentences is statistically insignificant. The only exception is the Stanford bidirectional tagger,

[6] Rescoring using the CRF model was also performed, but led to a performance degradation. We believe this is due to the fact that the prosodically enriched CRF model was able to directly use the break index information, and so restricting it to the baseline CRF model search space limits the performance to that of the baseline model.



(a) Number of errors by part of speech category for the HMM-LA model with and without prosody

(b) Number of errors by part of speech category for the CRF model with and without prosody

Figure 5: Error reduction for prosodically enriched HMM-LA (a) and CRF (b) models

which performs worse on these slightly longer segments. With the addition of break indexes, we see marginal changes in most of the models; only the CRF tagger receives a significant boost. Thus, models achieve performance gains from tagging shorter segments, but at the cost of limited usefulness of the prosodic breaks. Overall, speaker turn segmentation is an attractive compromise between the original conversation sides and human-annotated sentences.

6 Discussion

Across the different models, we have found that taggers applied to shorter segments, either sentences or speaker turns, do not tend to benefit significantly from prosodic enrichment, in contrast to conversation sides. To analyze this further we broke down the results by part of speech for the two models for which break indexes improved performance the most: the CRF and HMM-LA rescoring models, which achieved an overall error reduction of 2.8% and 2.1%, respectively. We present those categories that obtained the greatest benefit from prosody in Figure 5 (a) and (b). For both models, the UH category had a dramatic improvement from the addition of prosody, achieving up to a 10% reduction in error.

For the CRF model, other categories that saw impressive error reductions were NN and VB, with 10% and 5%, respectively. Table 5 lists the prosodic features that received the highest weight in the CRF model. These are quite intuitive, as they seem to represent places where the prosody indicates sentence or clausal boundaries. For the HMM-LA model, the VB and DT tags had major reductions in error of 13% and 10%, respectively. For almost all categories, the number of errors is reduced by the addition of breaks, and further reduced by using the OracleBreak processing described above.

Weight	**Feature**
2.2212	w_i=um & b_i=4 & t=UH
1.9464	w_i=uh & b_i=4 & t=UH
1.7965	w_i=yes & b_i=4 & t=UH
1.7751	w_i=and & b_i=4 & t=CC
1.7554	w_i=so & b_i=4 & t=RB
1.7373	w_i=but & b_i=4 & t=CC

Table 5: Top break 4 prosody features in CRF prosody model

To determine more precisely the effect that the segment size has on tagging accuracy, we extracted the oracle tag sequences from the HMM-LA and CRF baseline and prosodically enriched models across conversation sides, sentences, and speaker turn segments. As the plot in Figure 6 shows, as we increase the n-best list size to 500, the oracle accuracy of the models trained on sentences in-

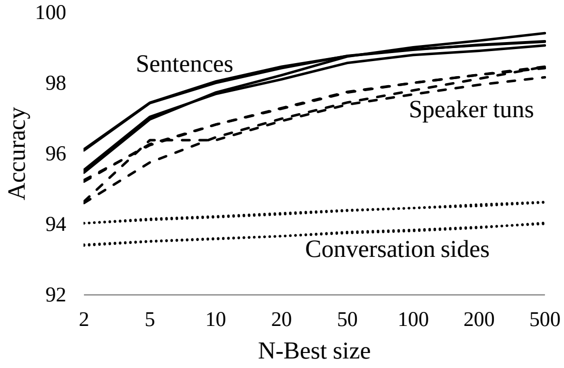


Figure 6: Oracle comparison: solid lines for sentences, dashed lines for speaker turns, and dotted lines for conversation sides

creases rapidly to 99%; whereas, the oracle accuracy of models on conversation sides grow slowly to between 94% and 95%. The speaker turn trained models, however, behave closely to those using sentences, climbing rapidly to accuracies of around 98%. This difference is directly attributable to the length of the segments. As can be seen in Table 6, the speaker turn segments are more comparable in length to sentences.

	Train	**Eval**
Conv	627.87 ± 281.57	502.98 ± 151.22
Sent	7.52± 7.86	7.45 ± 8.29
Speaker	15.60± 29.66	15.27± 21.01

Table 6: Length statistics of different data segmentations

Next, we return to the large performance degradation when tagging speech rather than newswire text to examine the major differences among the models. Using two of our best performing models, the Stanford bidirectional and HMM-LA, in Figure 7 we present the categories for which performance degradation was the greatest when comparing performance of a tagger trained on WSJ to a tagger trained on spoken sentences and conversation sides. The performance decrease is quite similar across both models, with the greatest degradation on the NNP, RP, VBN, and RBS categories.

Unsurprisingly, both the discriminative and generative bidirectional models achieve the most impressive results. However, the generative HMM-LA and HMM-LA-Bidir models achieved the best results across all three segmentations, and the best overall result, of 94.35%, on prosodically enriched sentence-segmented data. Since the Stanford bidirectional model incorporates all of the features that produced its state-of-the-art performance on WSJ, we believe the fact that the HMM-LA outperforms it, despite the discriminative model's more expressive feature set, is indicative of the HMM-LA's ability to more effectively adapt to novel domains during training. Another challenge for the discriminative models is the need for regularization tuning, requiring additional time and effort to train several models and select the most appropriate parameter each time the domain changes. Whereas for the HMM-LA models, although we also train several models, they can be combined into a product model, such as that described by Petrov (2010), in order to further improve performance.

Since the prosodic breaks are noisier features than the others incorporated in the discriminative models, it may be useful to set their regularization parameter separately from the rest of the features, however, we have not explored this alternative. Our experiments used human transcriptions of the conversational speech; however, realistically our models would be applied to speech recognition transcripts. In such a case, word error will introduce noise in addition to the prosodic breaks. In future work, we will evaluate the use of break indexes for tagging when there is lexical error. We would also apply the n-best rescoring method to exploit break indexes in the HMM-LA bidirectional model, as this would likely produce further improvements.

7 Conclusion

In this work, we have evaluated factors that are important for developing accurate tagging models for speech. Given that prosodic breaks were effective knowledge sources for parsing, an important goal of this work was to evaluate their impact on various tagging model configurations. Specifically, we have examined the use of prosodic information for tagging conversational speech with several different discriminative and generative models across three different speech transcript segmentations. Our find-

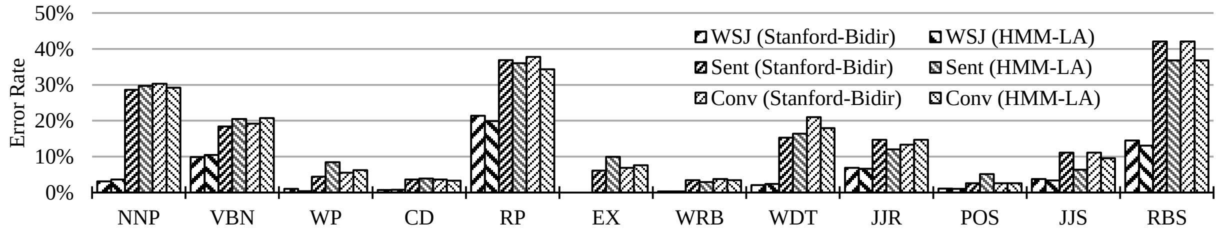


Figure 7: Comparison of error rates between the Standford Bidir and HMM-LA models trained on WSJ, sentences, and conversation sides

ings suggest that generative models with latent annotations achieve the best performance in this challenging domain. In terms of transcript segmentation, if sentences are available, it is preferable to use them. In the case that no such annotation is available, then using automatic sentence boundary detection does not serve as an appropriate replacement, but if automatic speaker turn segments can be obtained, then this is a good alternative, despite the fact that prosodic enrichment is less effective.

Our investigation also shows that in the event that conversation sides must be used, prosodic enrichment of the discriminative and generative models produces significant improvements in tagging accuracy (by direct integration of prosody features for the former and by restricting the search space and rescoring with the latter). For tagging, the most important role of the break indexes appears to be as a stand in for sentence boundaries. The oracle break experiments suggest that if the accuracy of the automatically induced break indexes can be improved, then the prosodically enriched models will perform as well, or even better, than their human-annotated sentence counterparts.

8 Acknowledgments

This research was supported in part by NSF IIS-0703859 and the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-2-001. Any opinions, findings, and recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agency or the institutions where the work was completed.

References

Anton Batliner, Bernd Möbius, Gregor Möhler, Antje Schweitzer, and Elmar Nöth. 2001. Prosodic models, automatic speech understanding, and speech synthesis: toward the common ground. In *Eurospeech*.

Ann Bies, Stephanie Strassel, Haejoong Lee, Kazuaki Maeda, Seth Kulick, Yang Liu, Mary Harper, and Matthew Lease. 2006. Linguistic resources for speech parsing. In *LREC*.

Stanley F. Chen and Ronald Rosenfeld. 1999. A Gaussian prior for smoothing maximum entropy models. Technical report, Technical Report CMU-CS-99-108, Carnegie Mellon University.

Anne Cutler, Delphine Dahan, and Wilma v an Donselaar. 1997. Prosody in comprehension of spoken language: A literature review. *Language and Speech*.

Markus Dreyer and Izhak Shafran. 2007. Exploiting prosody for PCFGs with latent annotations. In *Interspeech*.

Denis Filimonov and Mary Harper. 2009. A joint language model with fine-grain syntactic tags. In *EMNLP*.

Jennifer Foster. 2010. "cba to check the spelling": Investigating parser performance on discussion forum posts. In *NAACL-HLT*.

Florian Gallwitz, Heinrich Niemann, Elmar Nöth, and Volker Warnke. 2002. Integrated recognition of words and prosodic phrase boundaries. *Speech Communication*.

John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. SWITCHBOARD: Telephone speech corpus for research and development. In *ICASSP*.

Michelle L. Gregory, Mark Johnson, and Eugene Charniak. 2004. Sentence-internal prosody does not help parsing the way punctuation does. In *NAACL*.

Mary P. Harper, Bonnie J. Dorr, John Hale, Brian Roark, Izhak Shafran, Matthew Lease, Yang Liu, Matthew Snover, Lisa Yung, Anna Krasnyanskaya, and Robin Stewart. 2005. 2005 Johns Hopkins Summer Workshop Final Report on Parsing and Spoken Structural

Event Detection. Technical report, Johns Hopkins University.

Mark Hasegawa-Johnson, Ken Chen, Jennifer Cole, Sarah Borys, Sung suk Kim, Aaron Cohen, Tong Zhang, Jeung yoon Choi, Heejin Kim, Taejin Yoon, and Ra Chavarria. 2005. Simultaneous recognition of words and prosody in the boston university radio speech corpus. speech communication. *Speech Communication*.

Peter A. Heeman. 1999. POS tags and decision trees for language modeling. In *EMNLP*.

Dustin Hillard, Zhongqiang Huang, Heng Ji, Ralph Grishman, Dilek Hakkani-Tur, Mary Harper, Mari Ostendorf, and Wen Wang. 2006. Impact of automatic comma prediction on POS/name tagging of speech. In *ICASSP*.

Zhongqiang Huang and Mary Harper. 2010. Appropriately handled prosodic breaks help PCFG parsing. In *NAACL*.

Zhongqiang Huang, Vladimir Eidelman, and Mary Harper. 2009. Improving a simple bigram hmm part-of-speech tagger by latent annotation and self-training. In *NAACL-HLT*.

Jeremy G. Kahn, Matthew Lease, Eugene Charniak, Mark Johnson, and Mari Ostendorf. 2005. Effective use of prosody in parsing conversational speech. In *EMNLP-HLT*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

D. C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*.

Yang Liu, Andreas Stolcke, Elizabeth Shriberg, and Mary Harper. 2005. Using conditional random fields for sentence boundary detection in speech. In *ACL*.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *ACL*.

Mari Ostendorf, Izhak Shafran, and Rebecca Bates. 2003. Prosody models for conversational speech recognition. In *Plenary Meeting and Symposium on Prosody and Speech Processing*.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *HLT-NAACL*.

Slav Petrov. 2010. Products of random latent variable grammars. In *HLT-NAACL*.

Brian Roark, Yang Liu, Mary Harper, Robin Stewart, Matthew Lease, Matthew Snover, Izhak Shafran, Bonnie Dorr, John Hale, Anna Krasnyanskaya, and Lisa Yung. 2006. Reranking for sentence boundary detection in conversational speech. In *ICASSP*.

Kim Silverman, Mary Beckman, John Pitrelli, Mari Ostendorf, Colin Wightman, Patti Price, Janet Pierrehumbert, and Julia Hirshberg. 1992. ToBI: A standard for labeling English prosody. In *ICSLP*.

Paul Taylor and Alan W. Black. 1998. Assigning phrase breaks from part-of-speech sequences. *Computer Speech and Language*.

Scott M. Thede and Mary P. Harper. 1999. A second-order hidden markov model for part-of-speech tagging. In *ACL*.

Kristina Toutanova and Christopher D. Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *EMNLP*.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL*.

An Efficient Algorithm for Unsupervised Word Segmentation with Branching Entropy and MDL

Valentin Zhikov
Interdisciplinary Graduate School
of Science and Engineering
Tokyo Institute of Technology
zhikov@lr.pi.titech.ac.jp

Hiroya Takamura
Precision and Intelligence Laboratory
Tokyo Institute of Technology
takamura@pi.titech.ac.jp

Manabu Okumura
Precision and Intelligence Laboratory
Tokyo Institute of Technology
oku@pi.titech.ac.jp

Abstract

This paper proposes a fast and simple unsupervised word segmentation algorithm that utilizes the local predictability of adjacent character sequences, while searching for a least-effort representation of the data. The model uses branching entropy as a means of constraining the hypothesis space, in order to efficiently obtain a solution that minimizes the length of a two-part MDL code. An evaluation with corpora in Japanese, Thai, English, and the ”CHILDES” corpus for research in language development reveals that the algorithm achieves an accuracy, comparable to that of the state-of-the-art methods in unsupervised word segmentation, in a significantly reduced computational time.

1 Introduction

As an inherent preprocessing step to nearly all NLP tasks for writing systems without orthographical marking of word boundaries, such as Japanese and Chinese, the importance of word segmentation has lead to the emergence of a micro-genre in NLP focused exclusively on this problem.

Supervised probabilistic models such as Conditional Random Fields (CRF) (Lafferty et al., 2001) have a wide application to the morphological analysis of these languages. However, the development of the annotated training corpora necessary for their functioning is a labor-intensive task, which involves multiple stages of manual tagging. Because of the scarcity of labeled data, the domain adaptation of morphological analyzers is also problematic, and semi-supervised algorithms that address this issue have also been proposed (e.g. Liang, 2005; Tsuboi et al., 2008).

Recent advances in unsupervised word segmentation have been promoted by human cognition research, where it is involved in the modeling of the mechanisms that underlie language acquisition. Another motivation to study unsupervised approaches is their potential to support the domain adaptation of morphological analyzers through the incorporation of unannotated training data, thus reducing the dependency on costly manual work. Apart from the considerable difficulties in discovering reliable criteria for word induction, the practical application of such approaches is impeded by their prohibitive computational cost.

In this paper, we address the issue of achieving high accuracy in a practical computational time through an efficient method that relies on a combination of evidences: the local predictability of character patterns, and the reduction of effort achieved by a given representation of the language data. Both of these criteria are assumed to play a key role in native language acquisition. The proposed model allows experimentation in a more realistic setting, where the learner is able to apply them simultaneously. The

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 832–842,
MIT, Massachusetts, USA, 9-11 October 2010.

method shows a high performance in terms of accuracy and speed, can be applied to language samples of substantial length, and generalizes well to corpora in different languages.

2 Related Work

The principle of least effort (Zipf, 1949) postulates that the path of minimum resistance underlies all human behavior. Recent research has recognized its importance in the process of language acquisition (Kit, 2003). Compression-based word induction models comply to this principle, as they reorganize the data into a more compact representation while identifying the vocabulary of a text. The minimum description length framework (MDL) (Rissanen, 1978) is an appealing means of formalizing such models, as it provides a robust foundation for learning and inference, based solely on compression.

The major problem in MDL-based word segmentation is the lack of standardized search algorithms for the exponential hypothesis space (Goldwater, 2006). The representative MDL models compare favorably to the current state-of-the-art models in terms of accuracy. Brent and Cartwright (1996) carried out an exhaustive search through the possible segmentations of a limited subset of the data. Yu (2000) proposed an EM optimization routine, which achieved a high accuracy, in spite of a lower compression than the gold standard segmentation.

As a solution to the aforementioned issue, the proposed method incorporates the local predictability of character sequences into the inference process. Numerous studies have shown that local distributional cues can serve well the purpose of inducing word boundaries. Behavioral science has confirmed that infants are sensitive to the transitional probabilities found in speech (Saffran et al., 1996). The increase in uncertainty following a given word prefix is a well studied criterion for morpheme boundary prediction (Harris, 1955). A good deal of research has been conducted on methods through which such local statistics can be applied to the word induction problem (e.g. Kempe, 1999; Huang and Powers, 2003; Jin and Tanaka-Ishii, 2006). Hutchens and Adler (1998) noticed that entropic chunking has the effect of reducing the perplexity of a text.

Most methods for unsupervised word segmentation based solely on local statistics presume a certain – albeit minimum – level of acquaintance with the target language. For instance, the model of Huang and Powers (2003) involves some parameters (Markov chain order, numerous threshold values) that allow its adaptation to the individuality of written Chinese. In comparison, the method proposed in this paper generalizes easily to a variety of languages and domains, and is less dependent on annotated development data.

The state-of-the-art in unsupervised word segmentation is represented by Bayesian models. Goldwater et al. (2006) justified the importance of context as a means of avoiding undersegmentation, through a method based on hierarchical Dirichlet processes. Mochihashi et al. (2009) proposed extensions to this method, which included a nested character model and an optimized inference procedure. Johnson and Goldwater (2009) have proposed a novel method based on adaptor grammars, whose accuracy surpasses the aforementioned methods by a large margin, when appropriate assumptions are made regarding the structural units of a language.

3 Proposed Method

3.1 Word segmentation with MDL

The proposed two-part code incorporates some extensions of models presented in related work, aimed at achieving a more precise estimation of the representation length. We first introduce the general two-part code, which consists of:

- the model, embodied by a codebook, i.e., a lexicon of unique word types $M = \{w_1, ..., w_{|M|}\}$,
- the source text D, obtained through encoding the corpus using the lexicon.

The total description length amounts to the number of bits necessary for simultaneous transmission of the codebook and the source text. Therefore, our objective is to minimize the combined description length of both terms:

$$L(D, M) = L(M) + L(D|M).$$

The description length of the data given M is calculated using the Shannon-Fano code:

$$L(D|M) = -\sum_{j=1}^{|M|} \#w_j \log_2 P(w_j),$$

where $\#w_j$ stands for the frequency of the word w_j in the text.

Different strategies have been proposed in the literature for the calculation of the codebook cost. A common technique in segmentation and morphology induction models is to calculate the product of the total length in characters of the lexicon and an estimate of the per-character entropy. In this way, both the probabilities and lengths of words are taken into consideration. The use of a constant value is an effective and easily computable approach, but it is far from precise. For instance, in Yu (2000) the average entropy per character is measured against the original corpus, but this model does not capture the effects of the word distributions on the observed character probabilities. For this reason, we propose a different method: the codebook is modeled as a separate Markov chain of characters.

A lexicon of characters M' is defined. The description length of the lexicon data D' given M' is then calculated as:

$$L(D'|M') = -\sum_{i=1}^{|C|} \#c_i \log_2 P(c_i),$$

where $\#c_i$ denotes the frequency of a character c_i in the lexicon of hypothesis M. The term $L(M')$ is constant for any choice of hypothesis, as is represents the character set of a corpus.

The total description length under the proposed model is thus calculated as:

$$L(M) + L(D|M) = L(M') + L(D'|M') + L(D|M) = \\ -\sum_{i=1}^{|C|} \#c_i \log_2 P(c_i) - \sum_{j=1}^{|M|} \#w_j \log_2 P(w_j) + O(1).$$

A rigorous definition should include two additional terms, $L(\theta|M)$ and $L(\theta'|M')$, which give the representation cost of the parameters of both models. The $L(\theta|M)$ can be calculated as:

$$L(\theta|M) = \frac{|M|-1}{2} * \log_2 S,$$

where $|M|-1$ gives the number of parameters (degrees of freedom), and S is the size of the dataset (the total length of the text in characters). The parametric complexity term is calculated in the same way for the lexicon. For a derivation of the above formula, refer to e.g. Li (1998).

MDL is closely related to Bayesian inference. Depending on the choice of a universal code, the two approaches can overlap, as is the case with the two-part code discussed in this paper. It can be shown that the model selection in our method is equivalent to a MAP inference, conducted under the assumption that the prior probability of a model decreases exponentially with its length (Goldwater, 2006). Thus, the task that we are trying to accomplish is to conduct a focused search through the hypothesis space that will allow us to obtain an approximation of the MAP solution in a reasonable time.

The MDL framework does not provide standard search algorithms for obtaining the hypotheses that minimize the description length. In the rest of this section, we will describe an efficient technique suitable for the word segmentation task.

3.2 Obtaining an initial hypothesis

First, a rough initial hypothesis is built by an algorithm that combines the branching entropy and MDL criteria.

Given a set $\mathcal{X}$, comprising all the characters found in a text, the entropy of branching at position k of the text is defined as:

$$H(X_k|x_{k-1}, ..., x_{k-n}) = \\ -\sum_{x \in \mathcal{X}} P(x|x_{k-1}, ..., x_{k-n}) \log_2 P(x|x_{k-1}, ..., x_{k-n}),$$

where x_k represents the character found at position k, and n is the order of the Markov model over characters. For brevity, hereafter we shall denote the observed sequence $\{x_{k-1}, ..., x_{k-n}\}$ as $\{x_{k-1:k-n}\}$.

The above definition is extended to combine the entropy estimates in the left-to-right and right-to-left directions, as this factor has reportedly improved performance figures for models based on branching entropy (Jin and Tanaka-Ishii, 2006). The estimates in both directions are summed up, yielding a single value per position:

$$\begin{aligned}&H'(X_{k;k-1}|x_{k-1:k-n};x_{k:k+n-1}) = \\ &-\sum_{x\in\mathcal{X}} P(x|x_{k-1:k-n})\log_2 P(x|x_{k-1:k-n}) \\ &-\sum_{x\in\mathcal{X}} P(x|x_{k:k+n-1})\log_2 P(x|x_{k:k+n-1}).\end{aligned}$$

Suffix arrays are employed during the collection of frequency statistics. For a character model of order n over a testing corpus of size t and a training corpus of size m, suffix arrays allow these to be acquired in $O(tn \log m)$ time. Faster implementations reduce the complexity to $O(t(n+\log m))$. For further discussion, see Manber and Myers (1991). During the experiments, we did not use the caching functionality provided by the suffix array library, but instead kept the statistics for the current iterative pass (n-gram order and direction) in a local table.

The chunking technique we adopt is to insert a boundary when the branching entropy measured in sequences of length n exceeds a certain threshold value ($H(X|x_{k-1:k-n}) > \beta$). Both n and β are fixed.

Within the described framework, the increase in context length n promotes precision and recall at first, but causes a performance degradation when the entropy estimates become unreliable due to the reduced frequencies of long strings. High threshold values produce a combination of high precision and low recall, while low values result in low precision and high recall.

Since the F-score curve obtained as decreasing values are assigned to the threshold is typically unimodal as in many applications of MDL, we employ a bisection search routine for the estimation of the threshold (Algorithm 1).

All positions of the dataset are sorted by their entropy values. At each iteration, at most two new hypotheses are built, and their description lengths are calculated in time linear to the data size. The computational complexity of the described routine is $O(t \log t)$, where t is the corpus length in characters.

The order of the Markov chain n used during the entropy calculation is the only input variable of the proposed model. Since different values perform the best across the various languages, the most appropriate settings can be obtained with the help of a small annotated corpus. However, the MDL objective also enables unsupervised optimization against a sufficiently large unlabeled dataset. The order that minimizes the description length of the data can be discovered in a few iterations of Algorithm 1 with increasing values of n, and it typically matches the optimal value of the parameter (Table 1).

Algorithm 1 Generates an initial hypothesis.

```
thresholds[] := sorted $H(X_k)$ values;
threshold := median of thresholds[];
step := length of thresholds[]/4;
direction := ascending;
minimum := $+\infty$;
while $step > 0$ do
    nextThreshold := thresholds[] value one step in last direction;
    DL = calculateDL(nextThreshold);
    if $DL < minimum$ then
        minimum:= DL; threshold := nextThreshold;
        step := step/2; continue;
    end if
    reverse direction;
    nextThreshold := thresholds[] value one step in last direction;
    if $DL < minimum$ then
        minimum:= DL; threshold := nextThreshold;
        step := step/2; continue;
    end if
    reverse direction;
    step := step/2;
end while
```

Corpus	[1]	[2]	[3]	[4]
CHILDES	394655.52	**367711.66**	368056.10	405264.53
Kyoto	1.291E+07	**1.289E+07**	1.398E+07	1.837E+07

Table 1: Length in bits of the solutions proposed by Algorithm 1 with respect to the character n-gram order.

Although an acceptable initial segmentation can be built using the described approach, it is possible to obtain higher accuracy with an extended model that takes into account the statistics of Markov chains from several orders during the entropy calculation. This can be done by summing up the entropy estimates, in the way introduced earlier for combining the values in both directions:

$$\begin{aligned}&H''(X_{k;k-1}|x_{k-1:k-n};x_{k:k+n-1}) = \\ &-\sum_{n=1}^{n_{max}}(\sum_{x\in\mathcal{X}} P(x|x_{k-1:k-n})\log_2 P(x|x_{k-1:k-n}) \\ &+\sum_{x\in\mathcal{X}} P(x|x_{k:k+n-1})\log_2 P(x|x_{k:k+n-1})),\end{aligned}$$

where n_{max} is the index of the highest order to be taken into consideration.

3.3 Refining the initial hypothesis

In the second phase of the proposed method, we will refine the initial hypothesis through the reorganization of local co-occurrences which produce redundant description length. We opt for greedy optimization, as our primary interest is to further explore the impact that description length minimization has on accuracy. Of course, such an approach is unlikely to obtain global minima, but it is a feasible means of conducting the optimization process, and guarantees a certain increase in compression.

Since a preliminary segmentation is available, it is convenient to proceed by inserting or removing boundaries in the text, thus splitting or merging the already discovered tokens. The ranked positions involved in the previous step can be reused here, as this is a way to bias the search towards areas of the text where boundaries are more likely to occur. Boundary insertion should start in regions where the branching entropy is high, and removal should first occur in regions where the entropy is close to zero. A drawback of this approach is that it omits locations where the gains are not immediately obvious, as it cannot assess the cumulative gains arising from the merging or splitting of all occurrences of a certain pair (Algorithm 2).

A clean-up routine, which compensates for this shortage, is also implemented (Algorithm 3). It operates directly on the types found in the lexicon produced by Algorithm 2, and is capable of modifying a large number of occurrences of a given pair in a single step. The lexicon types are sorted by their contribution to the total description length of the corpus. For each word type, splitting or merging is attempted at every letter, beginning from the center. The algorithm eliminates unlikely types with low contribution, which represent mostly noise, and redistributes their cost among more likely ones. The design of the merging routine makes it impossible to produce types longer than the ones already found in the lexicon, as an exhaustive search would be prohibitive.

The evaluation of each hypothetical change in the segmentation requires that the description length of the two-part code is recalculated. In order to

Algorithm 2 Compresses local token co-occurrences.

```
path[][]:= positions sorted by $H(X_k)$ values;
minimum := DL of model produced at initialization;
repeat
    for i = max $H(X_k)$ to min $H(X_k)$ do
        pos:= path[i][k];
        if no boundary exists at pos then
            leftToken := token to the left;
            rightToken := token to the right;
            longToken := leftToken + rightToken;
            calculate DL after splitting;
            if $DL < minimum$ then
                accept split, update model, update DP variables;
            end if
        end if
    end for
    for i = min $H(X_k)$ to max $H(X_k)$ do
        merge leftToken and rightToken into longToken
        if DL will decrease (analogous to splitting)
    end for
until no change is evident in model
```

Algorithm 3 A lexicon clean-up procedure.

```
types[] := lexicon types sorted by cost;
minimum := DL of model produced by Algorithm 2;
repeat
    for i = min cost to max cost do
        for pos = middle to both ends of types[i] do
            longType := types[i];
            leftType := sequence from first character to pos;
            rightType:= sequence from pos to last character;
            calculate DL after splitting longType into leftType and rightType;
            if $DL < minimum$ then
                accept split, update model, update DP variables;
                break out of inner loop;
            end if
        end for
    end for
    types[] := lexicon types sorted by cost;
    for i = max cost to min cost do
        for pos = middle to both ends of types[i] do
            merge leftType and rightType into longType if
            DL will decrease (analogous to splitting)
            break out of inner loop;
        end for
    end for
until no change is evident in model
```

make this optimization phase computationally feasible, dynamic programming is employed in Algorithms 2 and 3. The approach adopted for the recalculation of the source text term $L(D|M)$ is explained below. The estimation of the lexicon cost is analogous. The term $L(D|M)$ can be rewritten as:

$$L(D|M) = -\sum_{j=1}^{|M|} \#w_j \log_2 \frac{\#w_j}{N} =$$

$$-\sum_{j=1}^{|M|} \#w_j \log_2 \#w_j + N \log_2 N = T_1 + T_2,$$

where $\#w_j$ is the frequency of w_j in the segmented corpus, and $N = \sum_{j=1}^{|M|} \#w_j$ is the cumulative token count. In order to calculate the new length, we keep the values of the terms T_1 and T_2 obtained at the last change of the model. Their new values are computed for each hypothetical split or merge on the basis of the last values, and the expected description length is calculated as their sum. If the produced estimate is lower, the model is modified and the new values of T_1 and T_2 are stored for future use.

In order to maintain precise token counts, Algorithms 2 and 3 recognize the fact that recurring sequences (”byebye” etc.) appear in the corpora, and handle them accordingly. Known boundaries, such as the sentence boundaries in the CHILDES corpus, are also taken into consideration.

4 Experimental Settings

We evaluated the proposed model against four datasets. The first one is *the Bernstein-Ratner corpus* for language acquisition based on transcripts from the CHILDES database (Bernstein-Ratner, 1987). It comprises phonetically transcribed utterances of adult speech directed to 13 through 21-month-old children. We evaluated the performance of our learner in the cases when the few boundaries among the individual sentences are available to it (B), and when it starts from a blank state (N). The *Kyoto University Corpus* (Kurohashi and Nagao, 1998) is a standard dataset for Japanese morphological and dependency structure analysis, which comprises newspaper articles and editorials from the Mainichi Shimbun. The *BEST* corpus for word segmentation and named entity recognition in Thai language combines text from a variety of sources including encyclopedias (E), newspaper articles (N), scientific articles (A), and novels (F). The *WSJ* subset of the *Penn Treebank II Corpus* incorporates selected stories from the Wall Street Journal, year 1989 (Marcus et al., 1994). Both the original text (O), and a version in which all characters were converted to lower case (L) were used.

Corpus	Language	Size (MB)	Chars (K)	Tokens (K)	Types (K)
CHILDES-B/N	English	0.1	95.8	33.3	1.3
Kyoto	Japanese	5.02	1674.9	972.9	39.5
WSJ	English	5.22	5220.0	1174.2	49.1
BEST-E	Thai	12.64	4360.2	1163.2	26.2
BEST-N	Thai	18.37	6422.7	1659.4	36.3
BEST-A	Thai	4.59	1619.9	438.7	13.9
BEST-F	Thai	16.18	5568.0	1670.8	22.6
Wikipedia	Japanese	425.0	169069.3	/	/
Asahi	Japanese	337.2	112401.1	/	/
BEST-All	Thai	51.2	17424.0	4371.8	73.4

Table 2: Corpora used during the evaluation. Precise token and type counts have been omitted for Wikipedia and Asahi, as no gold standard segmentations are available.

The datasets listed above were built by removing the tags and blank spaces found in the corpora, and concatenating the remaining text. We added two more training datasets for Japanese, which were used in a separate experiment solely for the acquisition of frequency statistics. One of them was created from 200,000 randomly chosen Wikipedia articles, stripped from structural elements. The other one contains text from the year 2005 issues of Asahi Newspaper. Statistics regarding all described datasets are presented in Table 2.

One whole corpus is segmented in each experiment, in order to avoid the statement of an extended model that would allow the separation of training and test data. This setting is also necessary for the direct comparison between the proposed model and other recent methods evaluated against the entire CHILDES corpus.

We report the obtained precision, recall and F-score values calculated using boundary, token and type counts. Precision (P) and recall (R) are defined as:

$$P = \frac{\#correct\ units}{\#\ output\ units}, \quad R = \frac{\#correct\ units}{\#gold\ standard\ units}.$$

Boundary, token and lexicon F-scores, denoted as B-F and T-F and L-F, are calculated as the

Model	Corpus & Settings	B-Prec	B-Rec	B-F	T-Prec	T-Rec	T-F	DL (bits)	Ref.DL (bits)	Time (ms)
1	CHILDES, α = 1.2, n = [1-6]	0.8667	0.8898	**0.8781**	**0.6808**	**0.6990**	**0.6898**	**344781.74**	300490.52	1060.2
2a (H')	CHILDES, n = 2	0.7636	**0.9109**	0.8308	0.5352	0.6384	0.5823	367711.66		**753.1**
2b (H'')	CHILDES, n_{max} = 3	**0.8692**	0.8865	0.8777	0.6792	0.6927	0.6859	347633.07		885.3
1	Kyoto, α = 0, n = [1-6]	**0.8208**	0.8208	0.8208	0.5784	0.5784	0.5784	1.325E+07	1.120E+07	54958.8
2a (H')	Kyoto, n = 2	0.8100	0.8621	0.8353	0.5934	0.6316	0.6119	1.289E+07		**22909.7**
2b (H'')	Kyoto, n_{max} = 2	0.8024	**0.9177**	**0.8562**	**0.6093**	**0.6969**	**0.6501**	**1.248+E07**		23212.8

Table 3: Comparison of the proposed method (2a, 2b) with the model of Jin and Tanaka-Ishii (2006) (1). Execution times include the obtaining of frequency statistics, and are represented by averages over 10 runs.

harmonic averages of the corresponding precision and recall values ($F = 2PR/(P + R)$). As a rule, boundary-based evaluation produces the highest scores among the three evaluation modes, as it only considers the correspondence between the proposed and the gold standard boundaries at the individual positions of the corpora. Token-based evaluation is more strict – it accepts a word as correct only if its beginning and end are identified accurately, and no additional boundaries lie in between. Lexicon-based evaluation reflects the extent to which the vocabulary of the original text has been recovered. It provides another useful perspective for the error analysis, which in combination with token scores can give a better idea of the relationship between the accuracy of induction and item frequency.

The system was implemented in Java, however it handled the suffix arrays through an external C library called Sary.[1] All experiments were conducted on a 2 GHz Core2Duo T7200 machine with 2 GB RAM.

5 Results and Discussion

The scores we obtained using the described instantiations of the branching entropy criterion at the initialization phase are presented in Table 3, along with those generated by our own implementation of the method presented in Jin and Tanaka-Ishii (2006), where the threshold parameter α was adjusted manually for optimal performance.

The heuristic of Jin and Tanaka-Ishii takes advantage of the trend that branching entropy decreases as the observed character sequences become longer; sudden rises can thus be regarded as an indication of locations where a boundary is likely to exist. Their method uses a common value for thresholding the entropy change throughout all n-gram orders, and combines the boundaries discovered in both directions in a separate step. These properties of the method would lead to complications if we tried to employ it in the first phase of our method (i.e. a step parameter for iterative adjustment of the threshold value, rules for combining the boundaries, etc.).

The proposed criterion with an automatically determined threshold value produced slightly worse results than that of Jin and Tanaka-Ishii at the CHILDES corpus. However, we found out that our approach achieves approximately 1% higher score when the best performing threshold value is selected from the candidate list. There are two observations that account for the suboptimal threshold choice by our algorithm. On one hand, the correspondence between description length and F-score is not absolutely perfect, and this may pose an obstacle to the optimization process for relatively small language samples. Another issue lies in the bisection search routine, which suggests approximations of the description length minima. The edge that our method has on the Kyoto corpus can be attributed to a better estimation of the optimal treshold value due to the larger amount of data.

The experimental results obtained at the completion of Algorithm 3 are summarized in Tables 4 and 5. Presented durations include the obtaining of frequency statistics. The n_{max} parameter is set to the value which maximizes the compression during the initial phase, in order to make the results representative of the case in which no annotated development corpora are accessible to the algorithm.

It is evident that after the optimization carried out in the second phase, the description length is reduced to levels significantly lower than the ground truth. In this aspect, the algorithm outperforms the EM-based method of Yu (2000).

[1] http://sary.sourceforge.net

Corpus & Settings	B-F	T-F	L-F	Time (ms)
CHILDES-B, n_{max}=3	0.9092	0.7542	0.5890	2597.2
CHILDES-N, n_{max}=3	0.9070	0.7499	0.5578	2949.3
Kyoto, n_{max}=2	0.8855	0.7131	0.3725	70164.6
BEST-E, n_{max}=5	0.9081	0.7793	0.3549	738055.0
BEST-N, n_{max}=5	0.8811	0.7339	0.2807	505327.0
BEST-A, n_{max}=5	0.9045	0.7632	0.4246	250863.0
BEST-F, n_{max}=5	0.9343	0.8216	0.4820	305522.0
WSJ-O, n_{max}=6	0.8405	0.6059	0.3338	658214.0
WSJ-L, n_{max}=6	0.8515	0.6373	0.3233	582382.0

Table 4: Results obtained after the termination of Algorithm 3.

Corpus & Settings	Description Length (Proposed)	Description Length (Total)
CHILDES-B, n_{max}=3	**290592.30**	300490.52
CHILDES-N, n_{max}=3	**290666.12**	300490.52
Kyoto, n_{max}=2	**1.078E+07**	1.120E+07
BEST-E, n_{max}=5	**1.180E+07**	1.252E+07
BEST-N, n_{max}=5	**1.670E+07**	1.809E+07
BEST-A, n_{max}=5	**4438600.32**	4711363.62
BEST-F, n_{max}=5	**1.562E+07**	1.634E+07
WSJ-O, n_{max}=6	**1.358E+07**	1.460E+07
WSJ-L, n_{max}=6	**1.317E+07**	1.399E+07

Table 5: Description length - proposed versus reference segmentation.

We conducted experiments involving various initialization strategies: scattering boundaries at random throughout the text, starting from entirely unsegmented state, or considering each symbol of the text to be a separate token. The results obtained with random initialization confirm the strong relationship between compression and segmentation accuracy, evident in the increase of token F-score between the random initialization and the termination of the algorithm, where description length is lower (Table 6). They also reveal the importance of the branching entropy criterion to the generation of hypotheses that maximize the evaluation scores and compression, as well as the role it plays in the reduction of computational time.

T-F-Score		Description	Time
Random Init	Refinement	Length	(ms)
0.0441 (0.25)	0.3833	387603.02	6660.4
0.0713 (0.50)	0.3721	383279.86	4975.1
0.0596 (0.75)	0.2777	412743.67	3753.3

Table 6: Experimental results for CHILDES-N with randomized initialization and search path. The numbers in brackets represent the seed boundaries/character ratios.

The greedy algorithms fail to suggest any optimizations that improve the compression in the extreme cases when the boundaries/character ratio is either 0 or 1. When no boundaries are given, splitting operations produce unique types with a low frequency that increase the cost of both parts of the MDL code, and are rejected. The algorithm runs slowly, as each evaluation operates on candidate strings of enormous length. Similarly, when the corpus is broken down into single-character tokens, merging individual pairs does not produce any increase in compression. This could be achieved by an algorithm that estimates the total effect from merging all instances of a given pair, but such an algorithm would be computationally infeasible for large corpora.

Finally, we tried randomizing the search path for Algorithm 2 after an entropy-guided initialization, to observe a small deterioration in accuracy in the final segmentation (less than 1% on average).

Figure 1a illustrates the effect that training data size has on the accuracy of segmentation for the Kyoto corpus. The learning curves are similar throughout the different corpora. For the CHILDES corpus, which has a rather limited vocabulary, token F-score above 70% can be achieved for datasets as small as 5000 characters of training data, provided that reasonable values are set for the n_{max} parameter (we used the values presented in Table 4 throughout these experiments).

Figure 1b shows the evolution of token F-score by stage for all corpora. The initialization phase seems to have the highest contribution to the formation of the final segmentation, and the refinement phase is highly dependent on the output it produces. As a consequence, results improve when a more adequate language sample is provided during the learning of local dependencies at initialization. This is evident in the experiments with the larger unlabeled Thai and Japanese corpora.

For Japanese language with the setting for the n_{max} parameter that maximized compression, we observed an almost 4% increase in the token F-score produced at the end of the first phase with the Asahi corpus as training data. Only a small (less than 1%) rise was observed in the overall performance. The quite larger dataset of randomly chosen Wikipedia articles achieved no improvement. We attributed this

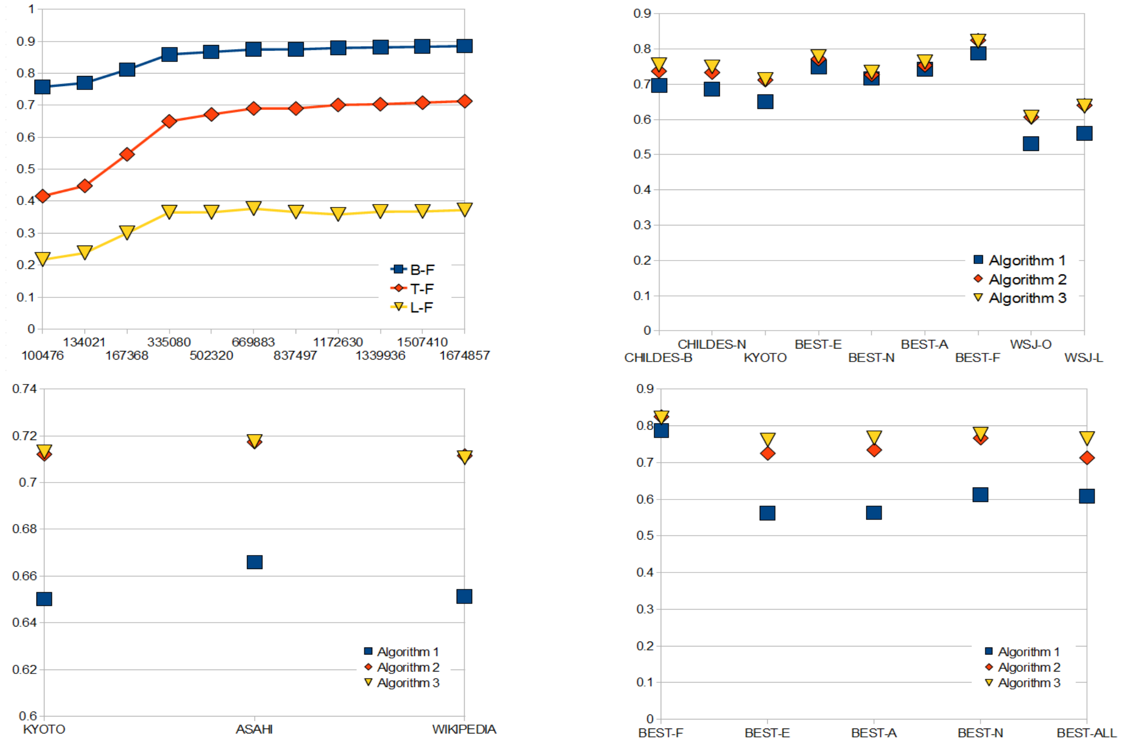


Figure 1: a) corpus size / accuracy relationship (Kyoto); b) accuracy levels by phase; c) accuracy levels by phase with various corpora for frequency statistics (Kyoto); d) accuracy levels by phase with different corpora for frequency statistics (BEST).

to the higher degree of correspondence between the domains of the Asahi and Kyoto corpora (Figure 1c).

Experiments with the BEST corpus reveal better the influence of domain-specific data on the accuracy of segmentation. Performance deteriorates significantly when out-of-domain training data is used. In spite of its size, the assorted composite corpus, in which in-domain and out-of-domain training data are mixed, produces worse results than the corpora which include only domain-specific data (Figure 1d).

Finally, a comparison of the proposed method with Bayesian n-gram models is presented in Table 7. Through the increase of compression in the refinement phase of the algorithm, accuracy is improved by around 3%, and the scores approach those of the explicit probabilistic models of Goldwater et al. (2009) and Mochihashi et al. (2009). The proposed learner surpasses the other unsupervised word induction models in terms of processing speed. It should be noticed that a direct comparison of accuracy is not possible with Mochihashi et al. (2009), as they evaluated their system with separate datasets for training and testing. Furthermore, different segmentation standards exist for Japanese, and therefore the ”ground truth” provided by the Kyoto corpus cannot be considered an ideal measure of accuracy.

6 Conclusions and Future Work

This paper has presented an efficient algorithm for unsupervised word induction, which relies on a combination of evidences. New instantiations of the branching entropy and MDL criteria have been proposed and evaluated against corpora in different languages. The MDL-based optimization eliminates the discretion in the choice of the context length and threshold parameters, common in segmentation models based on local statistics. At the same time, the branching entropy criterion enables a constrained search through the hypothesis space, allowing the proposed method to demonstrate a very high

Model	Corpus	T-Prec	T-Rec	T-F	L-Prec	L-Rec	L-F	Time
NPY(3)	CHILDES	0.7480	0.7520	0.7500	0.4780	**0.5970**	0.5310	17 min
NPY(2)	CHILDES	0.7480	**0.7670**	**0.7570**	0.5730	0.5660	0.5700	17 min
HDP(2)	CHILDES	0.7520	0.6960	0.7230	0.6350	0.5520	**0.5910**	-
Ent-MDL	CHILDES	**0.7634**	0.7453	0.7542	**0.6844**	0.5170	0.5890	**2.60 sec**
NPY(2)	Kyoto	-	-	0.6210	-	-	-	-
NPY(3)	Kyoto	-	-	0.6660	-	-	-	-
Ent-MDL	Kyoto	0.6912	0.7365	**0.7131**	0.5908	0.2720	0.3725	70.16 sec

Table 7: Comparison of the proposed method (Ent-MDL) with the methods of Mochihashi et al., 2009 (NPY) and Goldwater et al., 2009 (HDP).

performance in terms of both accuracy and speed.

Possible improvements of the proposed method include modeling the dependencies among neighboring tokens, which would allow the evaluation of the context to be reflected in the cost function. Mechanisms for stochastic optimization implemented in the place of the greedy algorithms could provide an additional flexibility of search for such more complex models. As the proposed approach provides significant performance improvements, it could be utilized in the development of more sophisticated novel word induction schemes, e.g. ensemble models trained independently with different data. Of course, we are also going to explore the model's potential in the setting of semi-supervised morphological analysis.

References

Bernstein-Ratner, Nan 1987. The phonology of parent – child speech. *Childrens Language*, 6:159–174

Brent, Michael R and Timothy A. Cartwright. 1996. Distributional Regularity and Phonotactic Constraints are Useful for Segmentation. *Cognition* 61: 93–125

Goldwater, Sharon. 2006. Nonparametric Bayesian Models of Lexical Acquisition. *Brown University*, Ph.D. Thesis

Goldwater, Sharon, Thomas L. Griffiths and Mark Johnson. 2006. Contextual dependencies in unsupervised word segmentation. *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, Sydney*, 673–680

Goldwater, Sharon, Thomas L. Griffiths and Mark Johnson. 2009. A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112:1, 21–54.

Harris, Zellig. 1955. From Phoneme to Morpheme. *Language*, 31(2):190-222.

Huang, Jin H. and David Powers. 2003. Chinese Word Segmentation Based on Contextual Entropy. *Proceedings of 17th Pacific Asia Conference*, 152–158

Hutchens, Jason L. and Michael D. Alder. 1998. Finding structure via compression. *Proceedings of the International Conference on Computational Natural Language Learning*, 79–82

Jin, Zhihui and Kumiko Tanaka-Ishii. 2006. Unsupervised Segmentation of Chinese Text by Use of Branching Entropy. *Proceedings of the COLING/ACL on Main conference poster sessions*, 428–435

Johnson, Mark and Sharon Goldwater. 2009. Improving nonparameteric Bayesian inference: experiments on unsupervised word segmentation with adaptor grammars. *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Association for Computational Linguistics*, 317–325.

Kempe, Andre. 1999. Experiments in Unsupervised Entropy Based Corpus Segmentation. *Proceedings of CoNLL'99*, pp. 371–385

Kit, Chunyu. 2003. How does lexical acquisition begin? A cognitive perspective. *Cognitive Science* 1(1): 1–50.

Kurohashi, Sadao and Makoto Nagao. 1998. Building a Japanese Parsed Corpus while Improving the Parsing System. *Proceedings of the First International Conference on Language Resources and Evaluation, Granada, Spain*, 719–724

Lafferty, John, Andrew McCallum and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. *Proceedings of the International Conference on Machine Learning*.

Li, Hang. 1998. A Probabilistic Approach to Lexical Semantic Knowledge Acquisition and Structural Disambiguation. *University of Tokyo*, Ph.D. Thesis

Liang, Percy. 2005. Semi-Supervised Learning for Natural Language. *Massachusets Institute of Technology*, Master's Thesis.

Manber, Udi and Gene Myers. 1991. Suffix arrays: a new method for on-line string searches. *SIAM Journal on Computing* 22:935–948

Marcus, Mitchell, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz and Britta Schasberger. 1994. The Penn Treebank: Annotating Predicate Argument Structure. *Human Language Technology*, 114–119

Mochihashi, Daiichi, Takeshi Yamada and Naonori Ueda. 2009. Bayesian unsupervised word segmentation with nested Pitman-Yor language modeling. *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, 1: 100–108

Rissanen, Jorma. 1978. Modeling by Shortest Data Description. *Aulomatica*, 14:465–471.

Saffran, Jenny R., Richard N. Aslin and Elissa L. Newport. 1996. Statistical learning in 8-month-old infants *Science*; 274:1926-1928

Tsuboi, Yuta, Hisashi Kashima., Hiroki Oda, Shinsuke Mori and Yuji Matsumoto. 2008. Training Conditional Random Fields Using Incomplete Annotations. *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*,897–904.

Yu, Hua. 2000. Unsupervised word induction using MDL criterion. *Proceedings of tne International Symposium of Chinese Spoken Language Processing, Beijing*.

Zipf, George K. 1949. Human Behavior and the Principle of Least Effort. *Addison-Wesley*.

A Fast Decoder for Joint Word Segmentation and POS-Tagging Using a Single Discriminative Model

Yue Zhang and **Stephen Clark**
University of Cambridge Computer Laboratory
William Gates Building,
15 JJ Thomson Avenue,
Cambridge CB3 0FD, UK
{yue.zhang, stephen.clark}@cl.cam.ac.uk

Abstract

We show that the standard beam-search algorithm can be used as an efficient decoder for the global linear model of Zhang and Clark (2008) for joint word segmentation and POS-tagging, achieving a significant speed improvement. Such decoding is enabled by: (1) separating full word features from partial word features so that feature templates can be instantiated incrementally, according to whether the current character is separated or appended; (2) deciding the POS-tag of a potential word when its first character is processed. Early-update is used with perceptron training so that the linear model gives a high score to a correct partial candidate as well as a full output. Effective scoring of partial structures allows the decoder to give high accuracy with a small beam-size of 16. In our 10-fold cross-validation experiments with the Chinese Treebank, our system performed over 10 times as fast as Zhang and Clark (2008) with little accuracy loss. The accuracy of our system on the standard CTB 5 test was competitive with the best in the literature.

1 Introduction and Motivation

Several approaches have been proposed to solve word segmentation and POS-tagging jointly, including the reranking approach (Shi and Wang, 2007; Jiang et al., 2008b), the hybrid approach (Nakagawa and Uchimoto, 2007; Jiang et al., 2008a), and the single-model approach (Ng and Low, 2004; Zhang and Clark, 2008; Kruengkrai et al., 2009). These methods led to accuracy improvements over the traditional, pipelined segmentation and POS-tagging baseline by avoiding segmentation error propagation and making use of part-of-speech information to improve segmentation.

The single-model approach to joint segmentation and POS-tagging offers consistent training of all information, concerning words, characters and parts-of-speech. However, exact inference with dynamic programming can be infeasible if features are defined over a large enough range of the output, such as over a two-word history. In our previous work (Zhang and Clark, 2008), which we refer to as Z&C08 from now on, we used an approximate decoding algorithm that keeps track of a set of partially built structures for each character, which can be seen as a dynamic programming chart which is greatly reduced by pruning.

In this paper we follow the line of single-model research, in particular the global linear model of Z&C08. We show that effective decoding can be achieved with standard beam-search, which gives significant speed improvements compared to the decoding algorithm of Z&C08, and achieves accuracies that are competitive with the state-of-the-art. Our research is also in line with recent research on improving the speed of NLP systems with little or no accuracy loss (Charniak et al., 2006; Roark and Hollingshead, 2008).

Our speed improvement is achieved by the use of a single-beam decoder. Given an input sentence, candidate outputs are built incrementally, one character at a time. When each character is processed, it is combined with existing candidates in all possible ways to generate new candidates, and an agenda is used to keep the N-best candidate outputs from

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 843–852,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

the begining of the sentence to the current character. Compared to the multiple-beam search algorithm of Z&C08, the use of a single beam can lead to an order of magnitude faster decoding speed.

1.1 The processing of partial words

An important problem that we solve in this paper is the handling of partial words with a single beam decoder for the global model. As we pointed out in Z&C08, it is very difficult to score partial words properly when they are compared with full words, although such comparison is necessary for incremental decoding with a single-beam. To allow comparisons with full words, partial words can either be treated as full words, or handled differently.

We showed in Z&C08 that a naive single-beam decoder which treats partial words in the same way as full words failed to give a competitive accuracy. An important reason for the low accuracy is over-segmentation during beam-search. Consider the three characters "自来水 (tap water)". The first two characters do not make sense when put together as a single word. Rather, when treated as two single-character words, they can make sense in a sentence such as "请 (please) 自 (self) 来 (come) 取 (take)". Therefore, when using single-beam search to process "自来水 (tap water)", the two-character word candidate "自来" is likely to have been thrown off the agenda before the third character "水" is considered, leading to an unrecoverable segmentation error.

This problem is even more severe for a joint segmentor and POS-tagger than for a pure word segmentor, since the POS-tags and POS-tag bigram of "自" and "来" further supports them being separated when "来" is considered. The multiple-beam search decoder we proposed in Z&C08 can be seen as a means to ensure that the three characters "自来水" always have a chance to be considered as a single word. It explores candidate segmentations from the beginning of the sentence until each character, and avoids the problem of processing partial words by considering only full words. However, since it explores a larger part of the search space than a single-beam decoder, its time complexity is correspondingly higher.

In this paper, we treat partial words differently from full words, so that in the previous example, the decoder can take the first two characters in "自来水 (tap water)" as a partial word, and keep it in the beam before the third character is processed. One challenge is the representation of POS-tags for partial words. The POS of a partial word is undefined without the corresponding full word information. Though a partial word can make sense with a particular POS-tag when it is treated as a complete word, this POS-tag is not necessarily the POS of the full word which contains the partial word. Take the three-character sequence "下雨天" as an example. The first character "下" represents a single-character word "below", for which the POS can be LC or VV. The first two characters "下雨" represent a two-character word "rain", for which the POS can be VV. Moreover, all three characters when put together make the word "rainy day", for which the POS is NN. As discussed above, assigning POS tags to partial words as if they were full words leads to low accuracy.

An obvious solution to the above problem is not to assign a POS to a partial word until it becomes a full word. However, lack of POS information for partial words makes them less competitive compared to full words in the beam, since the scores of full words are futher supported by POS and POS ngram information. Therefore, not assigning POS to partial words potentially leads to over segmentation. In our experiments, this method did not give comparable accuracies to our Z&C08 system.

In this paper, we take a different approach, and assign a POS-tag to a partial word when its first character is separated from the final character of the previous word. When more characters are appended to a partial word, the POS is not changed. The idea is to use the POS of a partial word as the predicted POS of the full word it will become. Possible predictions are made with the first character of the word, and the likely ones will be kept in the beam for the next processing steps. For example, with the three characters "下雨天", we try to keep two partial words (besides full words) in the beam when the first word "下" is processed, with the POS being VV and NN, respectively. The first POS predicts the two-character word "下雨"，and the second the three-character word "下雨天". Now when the second character is processed, we still need to maintain the possible POS NN in the agenda, which predicts the three-character word "下雨天".

As a main contribution of this paper, we show that the mechanism of predicting the POS at the first character gives competitive accuracy. This mechanism can be justified theoretically. Unlike alphabetical languages, each Chinese character represents some specific meanings. Given a character, it is natural for a human speaker to know immediately what types of words it can start. The allows the knowledge of possible POS-tags of words that a character can start, using information about the character from the training data. Moreover, the POS of the previous words to the current word are also useful in deciding possible POS for the word.[1]

The mechanism of first-character decision of POS also boosts the efficiency, since the enumeration of POS is unecessary when a character is appended to the end of an existing word. As a result, the complexity of each processing step is reduce by half compared to a method without POS prediction.

Finally, an intuitive way to represent the status of a partial word is using a flag explicitly, which means an early decision of the segmentation of the next incoming character. We take a simpler alternative approach, and treat every word as a partial word until the next incoming character is separated from the last character of this word. Before a word is confirmed as a full word, we only apply to it features that represent its current partial status, such as character bigrams, its starting character and its part-of-speech, etc. Full word features, including the first and last characters of a word, are applied immediately after a word is confirmed as complete.

An important component for our proposed system is the training process, which needs to ensure that the model scores a partial word with predicted POS properly. We use the averaged perceptron (Collins, 2002) for training, together with the "early update" mechanism of Collins and Roark (2004). Rather than updating the parameters after decoding is complete, the modified algorithm updates parameters at any processing step if the correct partial candidate falls out of the beam.

In our experiments using the Chinese Treebank data, our system ran an order of magnitude faster than our Z&C08 system with little loss of accuracy. The accuracy of our system was competitive with other recent models.

2 Model and Feature Templates

We use a linear model to score both partial and full candidate outputs. Given an input x, the score of a candidate output y is computed as:

$$\text{Score}(y) = \Phi(y) \cdot \vec{w},$$

where $\Phi(y)$ is the global feature vector extracted from y, and $\vec{w}$ is the parameter vector of the model.

Figure 1 shows the feature templates for the model, where templates 1 – 14 contain only segmentation information and templates 15 – 29 contain both segmentation and POS information. Each template is instantiated according to the current character in the decoding process. Row "For" shows the conditions for template instantiation, where "s" indicates that the corresponding template is instantiated when the current character starts a new word, and "a" indicates that the corresponding template is instantiated when the current character does not start a new word. In the row for feature templates, w, t and c are used to represent a word, a POS-tag and a character, respectively. The subscripts are based on the current character, where w_{-1} represents the first word to the left of the current character, and p_{-2} represents the POS-tag on the second word to the left of the current character, and so on. As an example, feature template 1 is instantiated when the current character starts a new word, and the resulting feature value is the word to the left of this character. $start(w)$, $end(w)$ and $len(w)$ represent the first character, the last character and the length of word w, respectively. The length of a word is normalized to 16 if it is larger than 16. $cat(c)$ represents the POS category of character c, which is the set of POS-tags seen on character c, as we used in Z&C08.

Given a partial or complete candidate y, its global feature vector $\Phi(y)$ is computed by instantiating all applicable feature templates from Table 1 for each character in y, according to whether or not the character is separated from the previous character.

The feature templates are mostly taken from, or inspired by, the feature templates of Z&C08. Templates 1, 2, 3, 4, 5, 8, 10, 12, 13, 14, 15, 19, 20,

[1] The next incoming characters are also a useful source of information for predicting the POS. However, our system achieved competitive accuracy with Z&C08 without such character lookahead features.

	Feature template	For
1	w_{-1}	s
2	$w_{-1}w_{-2}$	s
3	w_{-1}, where $len(w_{-1}) = 1$	s
4	$start(w_{-1})len(w_{-1})$	s
5	$end(w_{-1})len(w_{-1})$	s
6	$end(w_{-1})c_0$	s
7	$c_{-1}c_0$	a
8	$begin(w_{-1})end(w_{-1})$	s
9	$w_{-1}c_0$	s
10	$end(w_{-2})w_{-1}$	s
11	$start(w_{-1})c_0$	s
12	$end(w_{-2})end(w_{-1})$	s
13	$w_{-2}len(w_{-1})$	s
14	$len(w_{-2})w_{-1}$	s
15	$w_{-1}t_{-1}$	s
16	$t_{-1}t_0$	s
17	$t_{-2}t_{-1}t_0$	s
18	$w_{-1}t_0$	s
19	$t_{-2}w_{-1}$	s
20	$w_{-1}t_{-1}end(w_{-2})$	s
21	$w_{-1}t_{-1}c_0$	s
22	$c_{-2}c_{-1}c_0t_{-1}$, where $len(w_{-1}) = 1$	s
23	$start(w_0)t_0$	s
24	$t_{-1}start(w_{-1})$	s
25	t_0c_0	s, a
26	$c_0t_0start(w_0)$	a
27	$ct_{-1}end(w_{-1})$, where $c \in w_{-1}$ and $c \neq end(w_{-1})$	s
28	$c_0t_0cat(start(w_0))$	s
29	$ct_{-1}cat(end(w_{-1}))$, where $c \in w_{-1}$ and $c \neq end(w_{-1})$	s
30	$c_0t_0c_{-1}t_{-1}$	s
31	$c_0t_0c_{-1}$	a

Table 1: Feature templates.

24, 27 and 29 concern complete word information, and they are used in the model to differentiate correct and incorrect output structures in the same way as our Z&C08 model. Templates 6, 7, 9, 16, 17, 18, 21, 22, 23, 25, 26 and 28 concern partial word information, whose role in the model is to indicate the likelihood that the partial word including the current character will become a correct full word. They act as guidance for the action to take for the current character according to the context, and are the crucial reason for the effectiveness of the algorithm with a small beam-size.

```
function Decode(sent, agenda):
    Clear(agenda)
    AddItem(agenda, “”)
    for index in [0..Len(sent)]:
        for cand in agenda:
            new ← Append(cand, sent[index])
            AddItem(agenda, new)
            for pos in TagSet():
                new ← Sep(cand, sent[index], pos)
                AddItem(agenda, new)
        agenda ← N-Best(agenda)
    return Best(agenda)
```

Figure 1: The incremental beam-search decoder.

2.1 Decoding

The decoding algorithm builds an output candidate incrementally, one character at a time. Each character can either be attached to the current word or separated as the start a new word. When the current character starts a new word, a POS-tag is assigned to the new word. An agenda is used by the decoder to keep the N-best candidates during the incremental process. Before decoding starts, the agenda is initialized with an empty sentence. When a character is processed, existing candidates are removed from the agenda and extended with the current character in all possible ways, and the N-best newly generated candidates are put back onto the agenda. After all input characters have been processed, the highest-scored candidate from the agenda is taken as the output.

Pseudo code for the decoder is shown in Figure 1. CLEAR removes all items from the agenda, ADDITEM adds a new item onto the agenda, N-BEST returns the N highest-scored items from the agenda, and BEST returns the highest-scored item from the agenda. LEN returns the number of characters in a sentence, and *sent*[i] returns the ith character from the sentence. APPEND appends a character to the last word in a candidate, and SEP joins a character as the start of a new word in a candidate, assigning a POS-tag to the new word.

Both our decoding algorithm and the decoding algorithm of Z&C08 run in linear time. However, in order to generate possible candidates for each character, Z&C08 uses an extra loop to search for possible words that end with the current character. A restriction to the maximum word length is applied to limit the number of iterations in this loop, without which the algorithm would have quadratic time complexity. In contrast, our decoder does not search backword for the possible starting character of any word. Segmentation ambiguities are resolved by binary choices between the actions append or separate for each character, and no POS enumeration is required when the character is appended. This improves the speed by a significant factor.

2.2 Training

The learning algorithm is based on the generalized perceptron (Collins, 2002), but parameter adjustments can be performed at any character during the decoding process, using the "early update" mechanism of Collins and Roark (2004).

The parameter vector of the model is initialized as all zeros before training, and used to decode training examples. Each training example is turned into the raw input format, and processed in the same way as decoding. After each character is processed, partial candidates in the agenda are compared to the corresponding gold-standard output for the same characters. If none of the candidates in the agenda are correct, the decoding is stopped and the parameter vector is updated by adding the global feature vector of the gold-standard partial output and subtracting the global feature vector of the highest-scored partial candidate in the agenda. The training process then moves on to the next example. However, if any item in the agenda is the same as the corresponding gold-standard, the decoding process moves to the next character, without any change to the parameter values. After all characters are processed, the decoder prediction is compared with the training example. If the prediction is correct, the parameter vector is not changed; otherwise it is updated by adding the global feature vector of the training example and subtracting the global feature vector of the decoder prediction, just as the perceptron algorithm does. The same training examples can be used to train the model for multiple iterations. We use the averaged parameter vector (Collins, 2002) as the final model.

Pseudocode for the training algorithm is shown in Figure 2. It is based on the decoding algorithm in Figure 1, and the main differences are: (1) the training algorithm takes the gold-standard output and the parameter vector as two additional arguments; (2) the training algorithm does not return a prediction, but modifies the parameter vector when necessary; (3) lines 11 to 20 are additional lines of code for parameter updates.

Without lines 11 to 16, the training algorithm is exactly the same as the generalized perceptron algorithm. These lines are added to ensure that the agenda contains highly probable candidates during the whole beam-search process, and they are crucial to the high accuracy of the system. As stated earlier, the decoder relies on proper scoring of partial words to maintain a set of high quality candidates in the agenda. Updating the value of the parameter vector for partial outputs can be seen as a means to ensure correct scoring of partial candidates at any character.

2.3 Pruning

We follow Z&C08 and use several pruning methods, most of which serve to to improve the accuracy by removing irrelevant candidates from the beam. First, the system records the maximum number of characters that a word with a particular POS-tag can have. For example, from the Chinese Treebank that we used for our experiments, most POS are associated with only with one- or two-character words. The only POS-tags that are seen with words over ten characters long are NN (noun), NR (proper noun) and CD (numbers). The maximum word length information is initialized as all ones, and updated according to each training example before it is processed.

Second, a tag dictionary is used to record POS-tags associated with each word. During decoding, frequent words and words with "closed set" tags[2] are only allowed POS-tags according to the tag dictionary, while other words are allowed every POS-tag to make candidate outputs. Whether a word is a frequent word is decided by the number of times it has been seen in the training process. Denoting the num-

[2] "Closed set" tags are the set of POS-tags which are only associated with a fixed set of words, according to the Penn Chinese Treebank specifications (Xia, 2000).

```
function TRAIN(sent, agenda, gold-standard, $\vec{w}$):
    CLEAR(agenda)
    ADDITEM(agenda, “”)
    for index in [0..LEN(sent)]:
        for cand in agenda:
            new ← APPEND(cand, sent[index])
            ADDITEM(agenda, new)
            for pos in TAGSET():
                new ← SEP(cand, sent[index], pos)
                ADDITEM(agenda, new)
        agenda ← N-BEST(agenda)
        for cand in agenda:
            if cand = gold-standard[0:index]:
                CONTINUE
        $\vec{w}$ ← $\vec{w}$ + $\Phi$(gold-standard[0:index])
        $\vec{w}$ ← $\vec{w}$ - $\Phi$(BEST(agenda))
        return
    if BEST(agenda) $\neq$ gold-standard:
        $\vec{w}$ ← $\vec{w}$ + $\Phi$(gold-standard)
        $\vec{w}$ ← $\vec{w}$ - $\Phi$(BEST(agenda))
        return
    return
```

Figure 2: The incremental learning function.

ber of times the most frequent word has been seen with M, a word is a frequent word if it has been seen more than $M/5000 + 5$ times. The threshold value is taken from Z&C08, and we did not adjust it during development. Word frequencies are initialized as zeros and updated according to each training example before it is processed; the tag dictionary is initialized as empty and updated according to each training example before it is processed.

Third, we make an additional record of the initial characters for words with “closed set” tags. During decoding, when the current character is added as the start of a new word, “closed set” tags are only assigned to the word if it is consistent with the record. This type of pruning is used in addition to the tag dictionary to prune invalid partial words, while the tag dictionary is used to prune complete words. The record for initial character and POS is initially empty, and udpated according to each training example before it is processed.

Finally, at any decoding step, we group partial candidates that are generated by separating the current character as the start of a new word by the signature $p_0p_{-1}w_{-1}$, and keep only the best among those having the same $p_0p_{-1}w_{-1}$. The signature $p_0p_{-1}w_{-1}$ is decided by the feature templates we use: it can be shown that if two candidates *cand1* and *cand2* generated at the same step have the same signature, and the score of *cand1* is higher than the score of *cand2*, then at any future step, the highest scored candidate generated from *cand1* will always have a higher score than the highest scored candidate generated from *cand2*.

From the above pruning methods, only the third was not used by Z&C08. It can be seen as an extra mechanism to help keep likely partial words in the agenda and improve the accuracy, but which does not give our system a speed advantage over Z&C08.

3 Experiments

We used the Chinese Treebank (CTB) data to perform one set of development tests and two sets of fi-

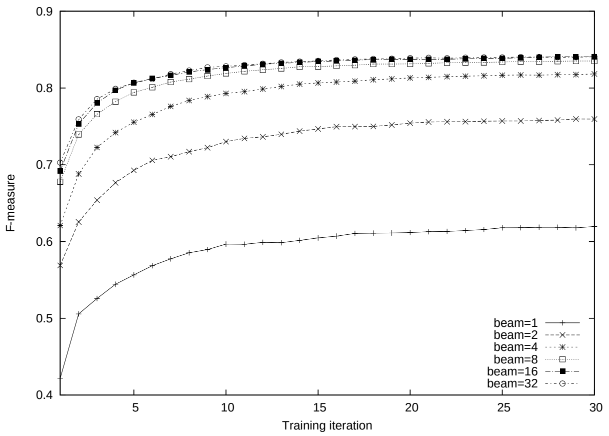


Figure 3: The influence of beam-sizes, and the convergence of the perceptron.

nal tests. The CTB 4 was split into two parts, with the CTB 3 being used for a 10-fold cross validation test to compare speed and accuracies with Z&C08, and the rest being used for development. The CTB 5 was used to perform the additional set of experiments to compare accuracies with other recent work.

We use the standard F-measure to evaluate output accuracies. For word segmentation, precision is defined as the number of correctly segmented words divided by the total number of words in the output, and recall is defined as the number of correctly segmented words divided by the total number of words in the gold-standard output. For joint segmentation and POS-tagging, precision is defined as the number of correctly segmented and POS-tagged words divided by the total number of words from the output, and recall is defined as the correctly segmented and POS-tagged words divided by the total number of words in the gold-standard output.

All our experiments were performed on a Linux platform, and a single 2.66GHz Intel Core 2 CPU.

3.1 Development tests

Our development data consists of 150K words in 4798 sentences. 80% of the data were randomly chosen as the development training data, while the rest were used as the development test data. Our development tests were mainly used to decide the size of the beam, the number of training iterations, the effect of partial features in beam-search decoding, and the effect of incremental learning (i.e. early update).

Figure 3 shows the accuracy curves for joint segmentation and POS-tagging by the number of training iterations, using different beam sizes. With the size of the beam increasing from 1 to 32, the accuracies generally increase, while the amount of increase becomes small when the size of the beam becomes 16. After the 10th iteration, a beam size of 32 does not always give better accuracies than a beam size of 16. We therefore chose 16 as the size of the beam for our system.

The testing times for each beam size between 1 and 32 are 7.16s, 11.90s, 18.42s, 27.82s, 46.77s and 89.21s, respectively. The corresponding speeds in the number of sentences per second are 111.45, 67.06, 43.32, 28.68, 17.06 and 8.95, respectively.

Figure 3 also shows that the accuracy increases with an increased number of training iterations, but the amount of increase becomes small after the 25th iteration. We chose 29 as the number of iterations to train our system.

The effect of incremental training: We compare the accuracies by incremental training using early update and normal perceptron training. In the normal perceptron training case, lines 11 to 16 are taken out of the training algorithm in Figure 2. The algorithm reached the best performance at the 22nd iteration, with the segmentation F-score being 90.58% and joint F-score being 83.38%. In the incremental training case, the algorithm reached the best accuracy at the 30th training iteration, obtaining a segmentation F-score of 91.14% and a joint F-score of 84.06%.

3.2 Final tests using CTB 3

CTB 3 consists of 150K words in 10364 sentences. We follow Z&C08 and split it into 10 equal-sized parts. In each test, one part is taken as the test data and the other nine are combined together as the training data. We compare the speed and accuracy with the joint segmentor and tagger of Z&C08, which is publicly available as the ZPar system, version 0.2[3].

The results are shown in Table 2, where each row shows one cross validation test. The column headings “sf”, “jf”, “time” and “speed” refer to segmentation F-measure, joint F-measure, testing time (in

[3] http://www.sourceforge.net/projects/zpar

	Z&C08				this paper			
#	sf	jf	time	speed	sf	jf	time	speed
1	97.18	93.27	557.97	1.86	97.25	93.51	44.20	23.44
2	97.65	93.81	521.63	1.99	97.66	93.97	42.07	24.26
3	96.08	91.04	444.69	2.33	95.55	90.65	39.23	26.41
4	96.31	91.93	431.04	2.40	96.37	92.15	39.54	26.20
5	96.35	91.94	508.39	2.04	95.84	91.51	43.30	23.93
6	94.48	88.63	482.78	2.15	94.25	88.53	43.77	23.67
7	95.27	90.52	361.95	2.86	95.10	90.42	41.76	24.81
8	94.98	90.01	418.54	2.47	94.87	90.30	39.81	26.02
9	95.23	90.84	471.3	2.20	95.21	90.55	42.03	26.65
10	96.49	92.11	500.72	2.08	96.33	92.12	43.12	24.03
average	96.00	91.41	469.90	2.24	95.84	91.37	41.88	24.94

Table 2: Speed and acccuracy comparisons with Z&C08 by 10-fold cross validation.

seconds) and testing speed (in the number of sentences per second), respectively.

Our system gave a joint segmentation and POS-tagging F-score of 91.37%, which is only 0.04% lower than that of ZPar 0.2. The speed of our system was over 10 times as fast as ZPar 0.2.

3.3 Final tests using CTB 5

We follow Kruengkrai et al. (2009) and split the CTB 5 into training, development testing and testing sets, as shown in Table 3. We ignored the development test data since our system had been developed in previous experiments.

	Sections	Sentences	Words
Training	1–270 400–931 1001–1151	18,085	493,892
Dev	301–325	350	6,821
Test	271–300	348	8,008

Table 3: Training, development and test data on CTB 5.

Kruengkrai et al. (2009) made use of character type knowledge for spaces, numerals, symbols, alphabets, Chinese and other characters. In the previous experiments, our system did not use any knowledge beyond the training data. To make the comparison fairer, we included knowledge of English letters and Arabic numbers in this experiment. During both training and decoding, English letters and Arabic numbers are segmented using simple rules, treating consecutive English letters or Arabic numbers as a single word.

The results are shown in Table 4, where row "N07" refers to the model of Nakagawa and Uchimoto (2007), rows "J08a" and "b" refer to the models of Jiang et al. (2008a) and Jiang et al. (2008b), and row "K09" refers to the models of Kruengkrai et al. (2009). Columns "sf" and "jf" refer to segmentation and joint accuracies, respectively. Our system gave comparable accuracies to these recent works, obtaining the best (same as the error-driven version of K09) joint F-score.

	sf	jf
K09 (error-driven)	97.87	93.67
our system	97.78	93.67
K09 (baseline)	97.79	93.60
J08a	97.85	93.41
J08b	97.74	93.37
N07	97.83	93.32

Table 4: Accuracy comparisons with recent studies on CTB 5.

4 Related Work

The effectiveness of our beam-search decoder showed that the joint segmentation and tagging problem may be less complex than previously perceived (Zhang and Clark, 2008; Jiang et al., 2008a). At the very least, the single model approach with a simple decoder achieved competitive accuracies to what has been achieved so far by the reranking (Shi and Wang, 2007; Jiang et al., 2008b) models and an ensemble model using machine-translation techniques (Jiang et al., 2008a). This may shed new light on joint segmentation and POS-tagging methods.

Kruengkrai et al. (2009) and Zhang and Clark (2008) are the most similar to our system among related work. Both systems use a discriminatively trained linear model to score candidate outputs. The work of Kruengkrai et al. (2009) is based on Nakagawa and Uchimoto (2007), which separates the processing of known words and unknown words, and uses a set of segmentation tags to represent the segmentation of characters. In contrast, our model is conceptually simpler, and does not differentiate known words and unknown words. Moreover, our model is based on our previous work, in line with Zhang and Clark (2007), which does not treat word segmentation as character sequence labeling.

Our learning and decoding algorithms are also different from Kruengkrai et al. (2009). While Kruengkrai et al. (2009) perform dynamic programming and MIRA learning, we use beam-search to perform incremental decoding, and the early-update version of the perceptron algorithm to train the model. Dynamic programming is exact inference, for which the time complexity is decided by the locality of feature templates. In contrast, beam-search is approximate and can run in linear time. The parameter updating for our algorithm is conceptually and computationally simpler than MIRA, though its performance can be slightly lower. However, the early-update mechanism we use is consistent with our incremental approach, and improves the learning of the beam-search process.

5 Conclusion

We showed that a simple beam-search decoding algorithm can be effectively applied to the decoding problem for a global linear model for joint word segmentation and POS-tagging. By guiding search with partial word information and performing learning for partial candidates, our system achieved significantly faster speed with little accuracy loss compared to the system of Z&C08.

The source code of our joint segmentor and POS-tagger can be found at:
www.sourceforge.net/projects/zpar, version 0.4.

Acknowledgements

We thank Canasai Kruengkrai for discussion on efficiency issues, and the anonymous reviewers for their suggestions. Yue Zhang and Stephen Clark are supported by the European Union Seventh Framework Programme (FP7-ICT-2009-4) under grant agreement no. 247762.

References

Eugene Charniak, Mark Johnson, Micha Elsner, Joseph Austerweil, David Ellis, Isaac Haxton, Catherine Hill, R. Shrivaths, Jeremy Moore, Michael Pozar, and Theresa Vu. 2006. Multilevel coarse-to-fine PCFG parsing. In *Proceedings of HLT/NAACL*, pages 168–175, New York City, USA, June. Association for Computational Linguistics.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL*, pages 111–118, Barcelona, Spain, July.

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, July.

Wenbin Jiang, Liang Huang, Qun Liu, and Yajuan Lü. 2008a. A cascaded linear model for joint Chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL/HLT*, pages 897–904, Columbus, Ohio, June.

Wenbin Jiang, Haitao Mi, and Qun Liu. 2008b. Word lattice reranking for Chinese word segmentation and part-of-speech tagging. In *Proceedings of COLING*, pages 385–392, Manchester, UK, August.

Canasai Kruengkrai, Kiyotaka Uchimoto, Jun'ichi Kazama, Yiou Wang, Kentaro Torisawa, and Hitoshi Isahara. 2009. An error-driven word-character hybrid model for joint Chinese word segmentation and POS tagging. In *Proceedings of ACL/AFNLP*, pages 513–521, Suntec, Singapore, August.

Tetsuji Nakagawa and Kiyotaka Uchimoto. 2007. A hybrid approach to word segmentation and POS tagging. In *Proceedings of ACL Demo and Poster Session*, Prague, Czech Republic, June.

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*, Barcelona, Spain.

Brian Roark and Kristy Hollingshead. 2008. Classifying chart cells for quadratic complexity context-free inference. In *Proceedings of COLING*, pages 745–752, Manchester, UK, August. Coling 2008 Organizing Committee.

Yanxin Shi and Mengqiu Wang. 2007. A dual-layer CRF based joint decoding method for cascade segmentation and labelling tasks. In *Proceedings of IJCAI*, Hyderabad, India.

Fei Xia, 2000. *The part-of-speech tagging guidelines for the Chinese Treebank (3.0)*.

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *Proceedings of ACL*, pages 840–847, Prague, Czech Republic, June.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and POS tagging using a single perceptron. In *Proceedings of ACL/HLT*, pages 888–896, Columbus, Ohio, June.

Simple Type-Level Unsupervised POS Tagging

Yoong Keok Lee Aria Haghighi Regina Barzilay
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{yklee, aria42, regina}@csail.mit.edu

Abstract

Part-of-speech (POS) tag distributions are known to exhibit *sparsity* — a word is likely to take a single predominant tag in a corpus. Recent research has demonstrated that incorporating this sparsity constraint improves tagging accuracy. However, in existing systems, this expansion come with a steep increase in model complexity. This paper proposes a simple and effective tagging method that directly models tag sparsity and other distributional properties of valid POS tag assignments. In addition, this formulation results in a dramatic reduction in the number of model parameters thereby, enabling unusually rapid training. Our experiments consistently demonstrate that this model architecture yields substantial performance gains over more complex tagging counterparts. On several languages, we report performance exceeding that of more complex state-of-the art systems.[1]

1 Introduction

Since the early days of statistical NLP, researchers have observed that a part-of-speech tag distribution exhibits "one tag per discourse" *sparsity* — words are likely to select a single predominant tag in a corpus, even when several tags are possible. Simply assigning to each word its most frequent associated tag in a corpus achieves 94.6% accuracy on the WSJ portion of the Penn Treebank. This distributional sparsity of syntactic tags is not unique to English — similar results have been observed across multiple languages. Clearly, explicitly modeling such a powerful constraint on tagging assignment has a potential to significantly improve the accuracy of an unsupervised part-of-speech tagger learned without a tagging dictionary.

In practice, this sparsity constraint is difficult to incorporate in a traditional POS induction system (Mérialdo, 1994; Johnson, 2007; Gao and Johnson, 2008; Graça et al., 2009; Berg-Kirkpatrick et al., 2010). These sequence models-based approaches commonly treat token-level tag assignment as the primary latent variable. By design, they readily capture regularities at the *token-level*. However, these approaches are ill-equipped to directly represent *type-based constraints* such as sparsity. Previous work has attempted to incorporate such constraints into token-level models via heavy-handed modifications to inference procedure and objective function (e.g., posterior regularization and ILP decoding) (Graça et al., 2009; Ravi and Knight, 2009). In most cases, however, these expansions come with a steep increase in model complexity, with respect to training procedure and inference time.

In this work, we take a more direct approach and treat a word type and its allowed POS tags as a primary element of the model. The model starts by generating a tag assignment for each word type in a vocabulary, assuming one tag per word. Then, token-level HMM emission parameters are drawn conditioned on these assignments such that each word is only allowed probability mass on a single assigned tag. In this way we restrict the parameterization of a

[1] The source code for the work presented in this paper is available at http://groups.csail.mit.edu/rbg/code/typetagging/.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 853–861,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

Language	Original case
English	94.6
Danish	96.3
Dutch	96.6
German	95.5
Spanish	95.4
Swedish	93.3
Portuguese	95.6

Table 1: Upper bound on tagging accuracy assuming each word type is assigned to majority POS tag. Across all languages, high performance can be attained by selecting a single tag per word type.

token-level HMM to reflect lexicon sparsity. This model admits a simple Gibbs sampling algorithm where the number of latent variables is proportional to the number of word types, rather than the size of a corpus as for a standard HMM sampler (Johnson, 2007).

There are two key benefits of this model architecture. First, it directly encodes linguistic intuitions about POS tag assignments: the model structure reflects the one-tag-per-word property, and a type-level tag prior captures the skew on tag assignments (e.g., there are fewer unique determiners than unique nouns). Second, the reduced number of hidden variables and parameters dramatically speeds up learning and inference.

We evaluate our model on seven languages exhibiting substantial syntactic variation. On several languages, we report performance exceeding that of state-of-the art systems. Our analysis identifies three key factors driving our performance gain: 1) selecting a model structure which directly encodes tag sparsity, 2) a type-level prior on tag assignments, and 3) a straightforward naïve-Bayes approach to incorporate features. The observed performance gains, coupled with the simplicity of model implementation, makes it a compelling alternative to existing more complex counterparts.

2 Related Work

Recent work has made significant progress on unsupervised POS tagging (Mérialdo, 1994; Smith and Eisner, 2005; Haghighi and Klein, 2006; Johnson, 2007; Goldwater and Griffiths, 2007; Gao and Johnson, 2008; Ravi and Knight, 2009). Our work is closely related to recent approaches that incorporate the sparsity constraint into the POS induction process. This line of work has been motivated by empirical findings that the standard EM-learned unsupervised HMM does not exhibit sufficient word tag sparsity.

The extent to which this constraint is enforced varies greatly across existing methods. On one end of the spectrum are clustering approaches that assign a single POS tag to each word type (Schutze, 1995; Lamar et al., 2010). These clusters are computed using an SVD variant without relying on transitional structure. While our method also enforces a singe tag per word constraint, it leverages the transition distribution encoded in an HMM, thereby benefiting from a richer representation of context.

Other approaches encode sparsity as a soft constraint. For instance, by altering the emission distribution parameters, Johnson (2007) encourages the model to put most of the probability mass on few tags. This design does not guarantee "structural zeros," but biases towards sparsity. A more forceful approach for encoding sparsity is posterior regularization, which constrains the posterior to have a small number of expected tag assignments (Graça et al., 2009). This approach makes the training objective more complex by adding linear constraints proportional to the number of word types, which is rather prohibitive. A more rigid mechanism for modeling sparsity is proposed by Ravi and Knight (2009), who minimize the size of tagging grammar as measured by the number of transition types. The use of ILP in learning the desired grammar significantly increases the computational complexity of this method.

In contrast to these approaches, our method directly incorporates these constraints into the structure of the model. This design leads to a significant reduction in the computational complexity of training and inference.

Another thread of relevant research has explored the use of features in unsupervised POS induction (Smith and Eisner, 2005; Berg-Kirkpatrick et al., 2010; Hasan and Ng, 2009). These methods demonstrated the benefits of incorporating linguistic features using a log-linear parameterization, but requires elaborate machinery for training. In our work, we demonstrate that using a simple naïve-Bayes approach also yields substantial performance gains, without the associated training complexity.

3 Generative Story

We consider the unsupervised POS induction problem without the use of a tagging dictionary. A graphical depiction of our model as well as a summary of random variables and parameters can be found in Figure 1. As is standard, we use a fixed constant K for the number of tagging states.

Model Overview The model starts by generating a tag assignment $\boldsymbol{T}$ for each word type in a vocabulary, assuming one tag per word. Conditioned on $\boldsymbol{T}$, features of word types $\boldsymbol{W}$ are drawn. We refer to $(\boldsymbol{T}, \boldsymbol{W})$ as the lexicon of a language and ψ for the parameters for their generation; ψ depends on a single hyperparameter β.

Once the lexicon has been drawn, the model proceeds similarly to the standard token-level HMM: Emission parameters θ are generated conditioned on tag assignments $\boldsymbol{T}$. We also draw transition parameters ϕ. Both parameters depend on a single hyperparameter α. Once HMM parameters (θ, ϕ) are drawn, a token-level tag and word sequence, (t, w), is generated in the standard HMM fashion: a tag sequence t is generated from ϕ. The corresponding token words w are drawn conditioned on t and θ.[2] Our full generative model is given by:

$$
\begin{aligned}
&P(\boldsymbol{T}, \boldsymbol{W}, \theta, \psi, \phi, \boldsymbol{t}, \boldsymbol{w} | \alpha, \beta) = \\
&\quad P(\boldsymbol{T}, \boldsymbol{W}, \psi | \beta) && \text{[Lexicon]} \\
&\quad P(\phi, \theta | \boldsymbol{T}, \alpha, \beta) && \text{[Parameter]} \\
&\quad P(\boldsymbol{w}, \boldsymbol{t} | \phi, \theta) && \text{[Token]}
\end{aligned}
$$

We refer to the components on the right hand side as the lexicon, parameter, and token component respectively. Since the parameter and token components will remain fixed throughout experiments, we briefly describe each.

Parameter Component As in the standard Bayesian HMM (Goldwater and Griffiths, 2007), all distributions are independently drawn from symmetric Dirichlet distributions:

$$P(\phi, \theta | \boldsymbol{T}, \alpha, \beta) = \prod_{t=1}^{K} \left(P(\phi_t | \alpha) P(\theta_t | \boldsymbol{T}, \alpha) \right)$$

The transition distribution ϕ_t for each tag t is drawn according to $\text{DIRICHLET}(\alpha, K)$, where α is the shared transition and emission distribution hyperparameter. In total there are $O(K^2)$ parameters associated with the transition parameters.

In contrast to the Bayesian HMM, θ_t is not drawn from a distribution which has support for each of the n word types. Instead, we condition on the type-level tag assignments $\boldsymbol{T}$. Specifically, let $S_t = \{i | T_i = t\}$ denote the indices of the word types which have been assigned tag t according to the tag assignments $\boldsymbol{T}$. Then θ_t is drawn from $\text{DIRICHLET}(\alpha, S_t)$, a symmetric Dirichlet which only places mass on word types indicated by S_t. This ensures that each word will only be assigned a single tag at inference time (see Section 4).

Note that while the standard HMM, has $O(Kn)$ emission parameters, our model has $O(n)$ effective parameters.[3]

Token Component Once HMM parameters (ϕ, θ) have been drawn, the HMM generates a token-level corpus $\boldsymbol{w}$ in the standard way:

$$P(\boldsymbol{w}, \boldsymbol{t} | \phi, \theta) = \prod_{(w,t) \in (\boldsymbol{w}, \boldsymbol{t})} \left(\prod_j P(t_j | \phi_{t_{j-1}}) P(w_j | t_j, \theta_{t_j}) \right)$$

Note that in our model, conditioned on $\boldsymbol{T}$, there is precisely one $\boldsymbol{t}$ which has non-zero probability for the token component, since for each word, exactly one θ_t has support.

3.1 Lexicon Component

We present several variations for the lexical component $P(\boldsymbol{T}, \boldsymbol{W} | \psi)$, each adding more complex parameterizations.

Uniform Tag Prior (1TW) Our initial lexicon component will be uniform over possible tag assignments as well as word types. Its only purpose is

[2] Note that t and w denote tag and word sequences respectively, rather than individual tokens or tags.

[3] This follows since each θ_t has $S_t - 1$ parameters and $\sum_t S_t = n$.

VARIABLES

$\boldsymbol{W}$: Word types $(W_1, \ldots, W_n)$ (obs)

$\boldsymbol{T}$: Tag assigns $(T_1, \ldots, T_n)$

$\boldsymbol{w}$: Token word seqs (obs)

$\boldsymbol{t}$: Token tag assigns (det by $\boldsymbol{T}$)

PARAMETERS

ψ: Lexicon parameters

θ : Token word emission parameters

ϕ : Token tag transition parameters

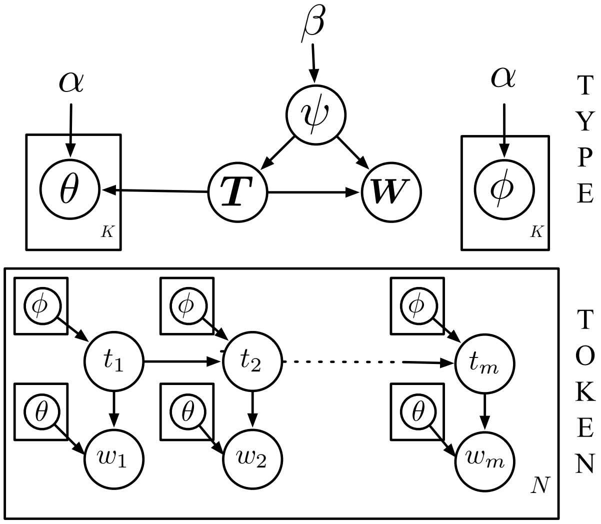


Figure 1: Graphical depiction of our model and summary of latent variables and parameters. The type-level tag assignments $\boldsymbol{T}$ generate features associated with word types $\boldsymbol{W}$. The tag assignments constrain the HMM emission parameters θ. The tokens $\boldsymbol{w}$ are generated by token-level tags $\boldsymbol{t}$ from an HMM parameterized by the lexicon structure. The hyperparameters α and β represent the concentration parameters of the token- and type-level components of the model respectively. They are set to fixed constants.

to explore how well we can induce POS tags using only the one-tag-per-word constraint. Specifically, the lexicon is generated as:

$$\begin{aligned} P(\boldsymbol{T}, \boldsymbol{W}|\psi) =& P(\boldsymbol{T})P(\boldsymbol{W}|\boldsymbol{T}) \\ =& \prod_{i=1}^{n} P(T_i)P(W_i|T_i) = \left(\frac{1}{Kn}\right)^n \end{aligned}$$

This model is equivalent to the standard HMM except that it enforces the one-word-per-tag constraint.

Learned Tag Prior (PRIOR) We next assume there exists a single prior distribution ψ over tag assignments drawn from $\textsc{Dirichlet}(\beta, K)$. This alters generation of $\boldsymbol{T}$ as follows:

$$P(\boldsymbol{T}|\psi) = \prod_{i=1}^{n} P(T_i|\psi)$$

Note that this distribution captures the frequency of a tag across word types, as opposed to tokens. The $P(T|\psi)$ distribution, in English for instance, should have very low mass for the DT (determiner) tag, since determiners are a very small portion of the vocabulary. In contrast, NNP (proper nouns) form a large portion of vocabulary. Note that these observations are not modeled by the standard HMM, which instead can model token-level frequency.

Word Type Features (FEATS): Past unsupervised POS work have derived benefits from features on word types, such as suffix and capitalization features (Hasan and Ng, 2009; Berg-Kirkpatrick et al., 2010). Past work however, has typically associated these features with token occurrences, typically in an HMM. In our model, we associate these features at the type-level in the lexicon. Here, we consider suffix features, capitalization features, punctuation, and digit features. While possible to utilize the feature-based log-linear approach described in Berg-Kirkpatrick et al. (2010), we adopt a simpler naïve Bayes strategy, where all features are emitted independently. Specifically, we assume each word type W consists of feature-value pairs (f, v). For each feature type f and tag t, a multinomial ψ_{tf} is drawn from a symmetric Dirichlet distribution with concentration parameter β. The $P(\boldsymbol{W}|\boldsymbol{T}, \psi)$ term in the lexicon component now decomposes as:

$$\begin{aligned} P(\boldsymbol{W}|\boldsymbol{T}, \psi) &= \prod_{i=1}^{n} P(W_i|T_i, \psi) \\ &= \prod_{i=1}^{n} \left(\prod_{(f,v) \in W_i} P(v|\psi_{T_i f}) \right) \end{aligned}$$

4 Learning and Inference

For inference, we are interested in the posterior probability over the latent variables in our model. During training, we treat as observed the language word types $\boldsymbol{W}$ as well as the token-level corpus $\boldsymbol{w}$. We utilize Gibbs sampling to approximate our collapsed model posterior:

$$P(\boldsymbol{T},\boldsymbol{t}|\boldsymbol{W},\boldsymbol{w},\alpha,\beta) \propto P(\boldsymbol{T},\boldsymbol{t},\boldsymbol{W},\boldsymbol{w}|\alpha,\beta)$$
$$= \int P(\boldsymbol{T},\boldsymbol{t},\boldsymbol{W},\boldsymbol{w},\psi,\theta,\phi,\boldsymbol{w}|\alpha,\beta)d\psi d\theta d\phi$$

Note that given tag assignments $\boldsymbol{T}$, there is only one setting of token-level tags $\boldsymbol{t}$ which has mass in the above posterior. Specifically, for the ith word type, the set of token-level tags associated with token occurrences of this word, denoted $\boldsymbol{t}^{(i)}$, must all take the value T_i to have non-zero mass. Thus in the context of Gibbs sampling, if we want to block sample T_i with $\boldsymbol{t}^{(i)}$, we only need sample values for T_i and consider this setting of $\boldsymbol{t}^{(i)}$.

The equation for sampling a single type-level assignment T_i is given by,

$$P(T_i,\boldsymbol{t}^{(i)}|\boldsymbol{T}_{-i},\boldsymbol{W},\boldsymbol{t}^{(-i)},\boldsymbol{w},\alpha,\beta) =$$
$$P(T_i|\boldsymbol{W},\boldsymbol{T}_{-i},\beta)P(\boldsymbol{t}^{(i)}|T_i,\boldsymbol{t}^{(-i)},\boldsymbol{w},\alpha)$$

where $\boldsymbol{T}_{-i}$ denotes all type-level tag assignment except T_i and $\boldsymbol{t}^{(-i)}$ denotes all token-level tags except $\boldsymbol{t}^{(i)}$. The terms on the right-hand-side denote the type-level and token-level probability terms respectively. The type-level posterior term can be computed according to,

$$P(T_i|\boldsymbol{W},\boldsymbol{T}_{-i},\beta) \propto$$
$$P(T_i|\boldsymbol{T}_{-i},\beta) \prod_{(f,v)\in W_i} P(v|T_i,f,\boldsymbol{W}_{-i},\boldsymbol{T}_{-i},\beta)$$

All of the probabilities on the right-hand-side are Dirichlet, distributions which can be computed analytically given counts.

The token-level term is similar to the standard HMM sampling equations found in Johnson (2007). The relevant variables are the set of token-level tags that appear before and after each instance of the ith word type; we denote these context pairs with the set $\{(t^b,t^a)\}$ and they are contained in $\boldsymbol{t}^{(-i)}$. We use w to represent the ith word type emitted by the HMM:

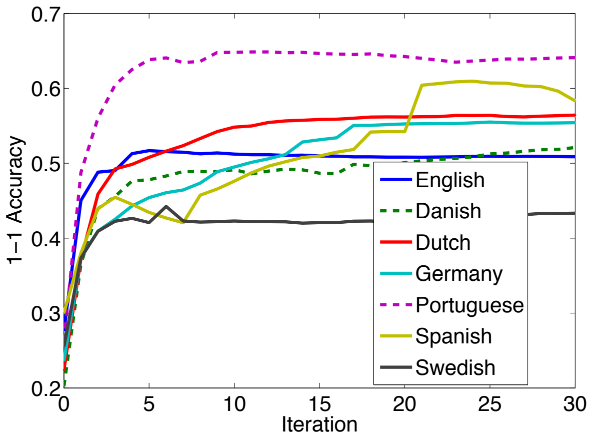


Figure 2: Graph of the one-to-one accuracy of our full model (+FEATS) under the best hyperparameter setting by iteration (see Section 5). Performance typically stabilizes across languages after only a few number of iterations.

$$P(\boldsymbol{t}^{(i)}|T_i,\boldsymbol{t}^{(-i)},\boldsymbol{w},\alpha) \propto$$
$$\prod_{(t^b,t^a)} P(w|T_i,\boldsymbol{t}^{(-i)},\boldsymbol{w}^{(-i)},\alpha)$$
$$P(T_i|t^b,\boldsymbol{t}^{(-i)},\alpha)P(t^a|T_i,\boldsymbol{t}^{(-i)},\alpha)$$

All terms are Dirichlet distributions and parameters can be analytically computed from counts in $\boldsymbol{t}^{(-i)}$ and $\boldsymbol{w}^{(-i)}$ (Johnson, 2007).

Note that each round of sampling T_i variables takes time proportional to the size of the corpus, as with the standard token-level HMM. A crucial difference is that the number of parameters is greatly reduced as is the number of variables that are sampled during each iteration. In contrast to results reported in Johnson (2007), we found that the performance of our Gibbs sampler on the basic 1TW model stabilized very quickly after about 10 full iterations of sampling (see Figure 2 for a depiction).

5 Experiments

We evaluate our approach on seven languages: English, Danish, Dutch, German, Portuguese, Spanish, and Swedish. On each language we investigate the contribution of each component of our model. For all languages we do not make use of a tagging dictionary.

Model	Hyper-	English		Danish		Dutch		German		Portuguese		Spanish		Swedish	
	param.	1-1	m-1	1-1	m-1	1-1	m-1	1-1	m-1	1-1	m-1	1-1	m-1	1-1	m-1
1TW	best	45.2	62.6	37.2	56.2	47.4	53.7	44.2	62.2	49.0	68.4	34.3	54.4	36.0	55.3
	median	45.1	61.7	32.1	53.8	43.9	61.0	39.3	68.4	48.5	68.1	33.6	54.3	34.9	50.2
+PRIOR	best	47.9	65.5	42.3	58.3	51.4	65.9	50.7	62.2	56.2	70.7	42.8	54.8	38.9	58.0
	median	46.5	64.7	40.0	57.3	48.3	60.7	41.7	68.3	52.0	70.9	37.1	55.8	36.8	57.3
+FEATS	best	50.9	66.4	52.1	61.2	56.4	69.0	55.4	70.4	64.1	74.5	58.3	68.9	43.3	61.7
	median	47.8	66.4	43.2	60.7	51.5	67.3	46.2	61.7	56.5	70.1	50.0	57.2	38.5	60.6

Table 3: Multi-lingual Results: We report token-level one-to-one and many-to-one accuracy on a variety of languages under several experimental settings (Section 5). For each language and setting, we report one-to-one (1-1) and many-to-one (m-1) accuracies. For each cell, the first row corresponds to the result using the best hyperparameter choice, where best is defined by the 1-1 metric. The second row represents the performance of the median hyperparameter setting. Model components cascade, so the row corresponding to +FEATS also includes the PRIOR component (see Section 3).

Language	# Tokens	# Word Types	# Tags
English	1173766	49206	45
Danish	94386	18356	25
Dutch	203568	28393	12
German	699605	72325	54
Portuguese	206678	28931	22
Spanish	89334	16458	47
Swedish	191467	20057	41

Table 2: Statistics for various corpora utilized in experiments. See Section 5. The English data comes from the WSJ portion of the Penn Treebank and the other languages from the training set of the CoNLL-X multilingual dependency parsing shared task.

5.1 Data Sets

Following the set-up of Johnson (2007), we use the whole of the Penn Treebank corpus for training and evaluation on English. For other languages, we use the CoNLL-X multilingual dependency parsing shared task corpora (Buchholz and Marsi, 2006) which include gold POS tags (used for evaluation). We train and test on the CoNLL-X training set. Statistics for all data sets are shown in Table 2.

5.2 Setup

Models To assess the marginal utility of each component of the model (see Section 3), we incrementally increase its sophistication. Specifically, we evaluate three variants: The first model (1TW) only encodes the one tag per word constraint and is uniform over type-level tag assignments. The second model (+PRIOR) utilizes the independent prior over type-level tag assignments $P(\boldsymbol{T}|\psi)$. The final model (+FEATS) utilizes the tag prior as well as features (e.g., suffixes and orthographic features), discussed in Section 3, for the $P(\boldsymbol{W}|\boldsymbol{T}, \psi)$ component.

Hyperparameters Our model has two Dirichlet concentration hyperparameters: α is the shared hyperparameter for the token-level HMM emission and transition distributions. β is the shared hyperparameter for the tag assignment prior and word feature multinomials. We experiment with four values for each hyperparameter resulting in 16 (α, β) combinations:

α	β
$0.001, 0.01, 0.1, 1.0$	$0.01, 0.1, 1.0, 10$

Iterations In each run, we performed 30 iterations of Gibbs sampling for the type assignment variables $\boldsymbol{W}$.[4] We use the final sample for evaluation.

Evaluation Metrics We report three metrics to evaluate tagging performance. As is standard, we report the *greedy one-to-one* (Haghighi and Klein, 2006) and the *many-to-one* token-level accuracy obtained from mapping model states to gold POS tags. We also report word type level accuracy, the fraction of word types assigned their majority tag (where the mapping between model state and tag is determined by greedy one-to-one mapping discussed above).[5]

For each language, we aggregate results in the following way: First, for each hyperparameter setting,

[4] Typically, the performance stabilizes after only 10 iterations.

[5] We choose these two metrics over the Variation Information measure due to the deficiencies discussed in Gao and Johnson (2008).

we perform five runs with different random initialization of sampling state. Hyperparameter settings are sorted according to the median one-to-one metric over runs. We report results for the best and median hyperparameter settings obtained in this way. Specifically, for both settings we report results on the median run for each setting.

Tag set As is standard, for all experiments, we set the number of latent model tag states to the size of the annotated tag set. The original tag set for the CoNLL-X Dutch data set consists of compounded tags that are used to tag multi-word units (MWUs) resulting in a tag set of over 300 tags. We tokenize MWUs and their POS tags; this reduces the tag set size to 12. See Table 2 for the tag set size of other languages. With the exception of the Dutch data set, no other processing is performed on the annotated tags.

6 Results and Analysis

We report token- and type-level accuracy in Table 3 and 6 for all languages and system settings. Our analysis and comparison focuses primarily on the one-to-one accuracy since it is a stricter metric than many-to-one accuracy, but also report many-to-one for completeness.

Comparison with state-of-the-art taggers For comparison we consider two unsupervised taggers: the HMM with log-linear features of Berg-Kirkpatrick et al. (2010) and the posterior regularization HMM of Graça et al. (2009). The system of Berg-Kirkpatrick et al. (2010) reports the best unsupervised results for English. We consider two variants of Berg-Kirkpatrick et al. (2010)'s richest model: optimized via either EM or LBFGS, as their relative performance depends on the language. Our model outperforms theirs on four out of five languages on the best hyperparameter setting and three out of five on the median setting, yielding an average absolute difference across languages of 12.9% and 3.9% for best and median settings respectively compared to their best EM or LBFGS performance. While Berg-Kirkpatrick et al. (2010) consistently outperforms ours on English, we obtain substantial gains across other languages. For instance, on Spanish, the absolute gap on median performance is 10%.

	Top 5	Bottom 5
Gold	NNP NN JJ CD NNS	RBS PDT # ” ,
1TW	**CD** WRB **NNS** VBN **NN**	PRP$ WDT : MD .
+PRIOR	**CD JJ NNS** WP$ **NN**	-RRB- **,** $ ” .
+FEATS	**JJ NNS CD NNP** UH	**,** PRP$ **#** . “

Table 5: Type-level English POS Tag Ranking: We list the top 5 and bottom 5 POS tags in the lexicon and the predictions of our models under the best hyperparameter setting.

Our second point of comparison is with Graça et al. (2009), who also incorporate a sparsity constraint, but does via altering the model objective using posterior regularization. We can only compare with Graça et al. (2009) on Portuguese (Graça et al. (2009) also report results on English, but on the reduced 17 tag set, which is not comparable to ours). Their best model yields 44.5% one-to-one accuracy, compared to our best median 56.5% result. However, our full model takes advantage of word features not present in Graça et al. (2009). Even without features, but still using the tag prior, our median result is 52.0%, still significantly outperforming Graça et al. (2009).

Ablation Analysis We evaluate the impact of incorporating various linguistic features into our model in Table 3. A novel element of our model is the ability to capture type-level tag frequencies. For this experiment, we compare our model with the uniform tag assignment prior (1TW) with the learned prior (+PRIOR). Across all languages, +PRIOR consistently outperforms 1TW, reducing error on average by 9.1% and 5.9% on best and median settings respectively. Similar behavior is observed when adding features. The difference between the featureless model (+PRIOR) and our full model (+FEATS) is 13.6% and 7.7% average error reduction on best and median settings respectively. Overall, the difference between our most basic model (1TW) and our full model (+FEATS) is 21.2% and 13.1% for the best and median settings respectively. One striking example is the error reduction for Spanish, which reduces error by 36.5% and 24.7% for the best and median settings respectively. We observe similar trends when using another measure – type-level accuracy (defined as the fraction of words correctly assigned their majority tag), according to which

Language	Metric	BK10 EM	BK10 LBFGS	G10	FEATS Best	FEATS Median
English	1-1	48.3	56.0	–	50.9	47.8
	m-1	68.1	75.5	–	66.4	66.4
Danish	1-1	42.3	42.6	–	52.1	43.2
	m-1	66.7	58.0	–	61.2	60.7
Dutch	1-1	53.7	55.1	–	56.4	51.5
	m-1	67.0	64.7	–	69.0	67.3
Portuguese	1-1	50.8	43.2	44.5	64.1	56.5
	m-1	75.3	74.8	69.2	74.5	70.1
Spanish	1-1	–	40.6	–	58.3	50.0
	m-1	–	73.2	–	68.9	57.2

Table 4: Comparison of our method (FEATS) to state-of-the-art methods. Feature-based HMM Model (Berg-Kirkpatrick et al., 2010): The KM model uses a variety of orthographic features and employs the EM or LBFGS optimization algorithm; Posterior regulariation model (Graça et al., 2009): The G10 model uses the posterior regularization approach to ensure tag sparsity constraint.

Language	1TW	+PRIOR	+FEATS
English	21.1	28.8	42.8
Danish	10.1	20.7	45.9
Dutch	23.8	32.3	44.3
German	12.8	35.2	60.6
Portuguese	18.4	29.6	61.5
Spanish	7.3	27.6	49.9
Swedish	8.9	14.2	33.9

Table 6: Type-level Results: Each cell report the type-level accuracy computed against the most frequent tag of each word type. The state-to-tag mapping is obtained from the best hyperparameter setting for 1-1 mapping shown in Table 3.

our full model yields 39.3% average error reduction across languages when compared to the basic configuration (1TW).

Table 5 provides insight into the behavior of different models in terms of the tagging lexicon they generate. The table shows that the lexicon tag frequency predicated by our full model are the closest to the gold standard.

7 Conclusion and Future Work

We have presented a method for unsupervised part-of-speech tagging that considers a word type and its allowed POS tags as a primary element of the model. This departure from the traditional token-based tagging approach allows us to explicitly capture type-level distributional properties of valid POS tag assignments as part of the model. The resulting model is compact, efficiently learnable and linguistically expressive. Our empirical results demonstrate that the type-based tagger rivals state-of-the-art tag-level taggers which employ more sophisticated learning mechanisms to exploit similar constraints.

In this paper, we make a simplifying assumption of one-tag-per-word. This assumption, however, is not inherent to type-based tagging models. A promising direction for future work is to explicitly model a distribution over tags for each word type. We hypothesize that modeling morphological information will greatly constrain the set of possible tags, thereby further refining the representation of the tag lexicon.

Acknowledgments

The authors acknowledge the support of the NSF (CAREER grant IIS-0448168, and grant IIS-0904684). We are especially grateful to Taylor Berg-Kirkpatrick for running additional experiments. We thank members of the MIT NLP group for their suggestions and comments. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of the funding organizations.

References

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proceedings of NAACL-HLT*, pages 582–590.

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *In Proc. of CoNLL*, pages 149–164.

Jianfeng Gao and Mark Johnson. 2008. A comparison of bayesian estimators for unsupervised hidden markov model pos taggers. In *Proceedings of the EMNLP*, pages 344–352.

Sharon Goldwater and Thomas L. Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the ACL*, pages 744–751.

João Graça, Kuzman Ganchev, Ben Taskar, and Fernando Pereira. 2009. Posterior vs. parameter sparsity in latent variable models. In *Proceeding of NIPS*, pages 664–672.

Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *Proceedings of the HLT-NAACL*, pages 320–327.

Kazi Saidul Hasan and Vincent Ng. 2009. Weakly supervised part-of-speech tagging for morphologically-rich, resource-scarce languages. In *Proceedings of EACL*, pages 363–371.

Mark Johnson. 2007. Why doesn't em find good hmm pos-taggers? In *Proceedings of EMNLP-CoNLL*, pages 296–305.

Michael Lamar, Yariv Maron, Marko Johnson, and Elie Bienstock. 2010. Svd Clustering for Unsupervised POS Tagging. In *Proceedings of ACL*, pages 215–219.

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

Sujith Ravi and Kevin Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *Proceedings of ACL-IJCNLP*, pages 504–512.

Hinrich Schutze. 1995. Distributional part of speech tagging. In *Proceedings of the EACL*, pages 141–148.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the ACL.*

Classifying Dialogue Acts in One-on-one Live Chats

Su Nam Kim,♠ Lawrence Cavedon♡ and Timothy Baldwin♠
♠ Dept of Computer Science and Software Engineering, University of Melbourne
♡ School of Computer Science and IT, RMIT University
sunamkim@gmail.com, lcavedon@gmail.com, tb@ldwin.net

Abstract

We explore the task of automatically classifying dialogue acts in 1-on-1 online chat forums, an increasingly popular means of providing customer service. In particular, we investigate the effectiveness of various features and machine learners for this task. While a simple bag-of-words approach provides a solid baseline, we find that adding information from dialogue structure and inter-utterance dependency provides some increase in performance; learners that account for sequential dependencies (CRFs) show the best performance. We report our results from testing using a corpus of chat dialogues derived from online shopping customer-feedback data.

1 Introduction

Recently, *live chats* have received attention due to the growing popularity of chat services and the increasing body of applications. For example, large organizations are increasingly providing support or information services through live chat. One advantage of chat-based customer service over conventional telephone-based customer service is that it becomes possible to semi-automate aspects of the interaction (e.g. conventional openings or canned responses to standard questions) without the customer being aware of it taking place, something that is not possible with speech-based dialogue systems (as synthesised speech is still easily distinguishable from natural speech). Potentially huge savings can be made by organisations providing customer help services if we can increase the degree of automation of live chat.

Given the increasing impact of live chat services, there is surprisingly little published computational linguistic research on the topic. There has been substantially more work done on dialogue and dialogue corpora, mostly in spoken dialogue (e.g. Stolcke et al. (2000)) but also multimodal dialogue systems in application areas such as telephone support service (Bangalore et al., 2006) and tutoring systems (Litman and Silliman, 2004). Spoken dialogue analysis introduces many complications related to the error inherent in current speech recognition technologies. As an instance of written dialogue, an advantage of live chats is that recognition errors are not such an issue, although the nature of language used in chat is typically ill-formed and turn-taking is complicated by the semi-asynchronous nature of the interaction (e.g. Werry (1996)).

In this paper, we investigate the task of automatic classification of *dialogue acts* in 1-on-1 live chats, focusing on "information delivery" chats since these are proving increasingly popular as part of enterprise customer-service solutions. Our main challenge is to develop effective features and classifiers for classifying aspects of 1-on-1 live chat. Much of the work on analysing dialogue acts in spoken dialogues has relied on non-lexical features, such as prosody and acoustic features (Stolcke et al., 2000; Julia and Iftekharuddin, 2008; Sridhar et al., 2009), which are not available for written dialogues. Previous dialogue-act detection for chat systems has used bags-of-words (hereafter, BoW) as features for dialogue-act detection; this simple approach has shown some promise (e.g. Bangalore et al. (2006), Louwerse and Crossley (2006) and Ivanovic (2008)). Other features such as keywords/ontologies (Purver et al., 2005; Forsyth, 2007) and lexical cues (Ang et al., 2005) have also been used for dialogue act classification.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 862–871,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

In this paper, we first re-examine BoW features for dialogue act classification. As a baseline, we use the work of Ivanovic (2008), which explored 1-grams and 2-grams with Boolean values in 1-on-1 live chats in the MSN Online Shopping domain (this dataset is described in Section 5). Although this work achieved reasonably high performance (up to a micro-averaged F-score of around 80%), we believe that there is still room for improvement using BoW only. We extend this work by using ideas from related research such as text categorization (Debole and Sebastiani, 2003), and explore variants of BoW based on analysis of live chats, along with feature weighting. Finally, our main aim is to explore new features based on dialogue structure and dependencies between utterances[1] that can enhance the use of BoW for dialogue act classification. Our hypothesis is that, for task-oriented 1-on-1 live chats, the structure and interactions among utterances are useful in predicting future dialogue acts: for example, conversations typically start with a greeting, and questions and answers typically appear as adjacency pairs in a conversation. Therefore, we propose new features based on structural and dependency information derived from utterances (Sections 4.2 and 4.3).

2 Related Work

While there has been significant work on classifying dialogue acts, the bulk of this has been for spoken dialogue. Most such work has considered: (1) defining taxonomies of dialogue acts; (2) discovering useful features for the classification task; and (3) experimenting with different machine learning techniques. We focus here on (2) and (3); we return to (1) in Section 3.

For classifying dialogue acts in spoken dialogue, various features such as dialogue cues, speech characteristics, and n-grams have been proposed. For example, Samuel et al. (1998) utilized the characteristics of spoken dialogues and examined speaker direction, punctuation marks, cue phrases and n-grams for classifying spoken dialogues. Jurafsky et al. (1998) used prosodic, lexical and syntactic features for spoken dialogue classification. More recently, Julia and Iftekharuddin (2008) and Sridhar et al. (2009) achieved high performance using acoustic and prosodic features. Louwerse and Crossley (2006), on the other hand, used various n-gram features—which could be adapted to both spoken and written dialogue—and tested them using the Map Task Corpus (Anderson et al., 1991). Extending the discourse model used in previous work, Bangalore et al. (2006) used n-grams from the previous 1–3 utterances in order to classify dialogue acts for the target utterance.

There has been substantially less effort on classifying dialogue acts in written dialogue: Wu et al. (2002) and Forsyth (2007) have used keyword-based approaches for classifying online chats; Ivanovic (2008) tested the use of n-gram features for 1-on-1 live chats with MSN Online Shopping assistants.

Various machine learning techniques have been investigated for the dialogue classification task. Samuel et al. (1998) used transformation-based learning to classify spoken dialogues, incorporating Monte Carlo sampling for training efficiency. Stolcke et al. (2000) used Hidden Markov Models (HMMs) to account for the structure of spoken dialogues, while Wu et al. (2002) also used transformation- and rule-based approaches plus HMMs for written dialogues. Other researchers have used Bayesian based approaches, such as naive Bayes (e.g. (Grau et al., 2004; Forsyth, 2007; Ivanovic, 2008)) and Bayesian networks (e.g. (Keizer, 2001; Forsyth, 2007)). Maximum entropy (e.g. (Ivanovic, 2008)), support vector machines (e.g. (Ivanovic, 2008)), and hidden Markov models (e.g. (Bui, 2003)) have also all been applied to automatic dialogue act classification.

3 Dialogue Acts

A number of dialogue act taxonomies have been proposed, designed mainly for spoken dialogue. Many of these use the Dialogue Act Markup in Several Layers (DAMSL) scheme (Allen and Core, 1997). DAMSL was originally applied to the TRAINS corpus of (transcribed) spoken task-oriented dialogues, but various adaptations of it have since been proposed for specific types of dialogue. The Switchboard corpus (Godfrey et al., 1992) defines 42 types of dialogue acts from human-to-human telephone conversations. The HCRC Map Task corpus (Ander-

[1] An *utterance* is the smallest unit to deliver a participant's message(s) in a turn.

son et al., 1991) defines a set of 128 dialogue acts to model task-based spoken conversations.

For casual online chat dialogues, Wu et al. (2002) define 15 dialogue act tags based on previously-defined dialogue act sets (Samuel et al., 1998; Shriberg et al., 1998; Jurafsky et al., 1998; Stolcke et al., 2000). Forsyth (2007) defines 15 dialogue acts for casual online conversations, based on 16 conversations with 10,567 utterances. Ivanovic (2008) proposes 12 dialogue acts based on DAMSL for 1-on-1 online customer service chats.

Ivanovic's set of dialogue acts for chat dialogues has significant overlap with the dialogue act sets of Wu et al. (2002) and Forsyth (2007) (e.g. GREETING, EMOTION/EXPRESSION, STATEMENT, QUESTION). In our work, we re-use the set of dialogue acts proposed in Ivanovic (2008), due to our targeting the same task of 1-on-1 IM chats, and indeed experimenting over the same dataset. The definitions of the dialogue acts are provided in Table 1, along with examples.

4 Feature Selection

In this section, we describe our initial dialogue-act classification experiments using simple BoW features, and then introduce two groups of new features based on structural information and dependencies between utterances.

4.1 Bag-of-Words

n-gram-based BoW features are simple yet effective for identifying similarities between two utterances, and have been used widely in previous work on dialogue act classification for online chat dialogues (Louwerse and Crossley, 2006; Ivanovic, 2008). However, chats containing large amounts of noise such as typos and emoticons pose a greater challenge for simple BoW approaches. On the other hand, keyword-based features (Forsyth, 2007) have achieved high performance; however, keyword-based approaches are more domain-dependent. In this work, we chose to start with a BoW approach based on our observation that commercial live chat services contain relatively less noise; in particular, the commercial agent tends to use well-formed, formulaic prose.

Previously, Ivanovic (2008) explored Boolean 1-gram and 2-gram features to classify MSN Online Shopping live chats, where a user requests assistance in purchasing an item, in response to which the commercial agent asks the customer questions and makes suggestions. Ivanovic (2008) achieved solid performance over this data (around 80% F-score). While 1-grams performed well (as live chat utterances are generally shorter than, e.g., sentences in news articles), we expect 2- and 3-grams are needed to detect formulaic expressions, such as *No problem* and *You are welcome*. We would also expect a positive effect from combining n-grams due to increasing the coverage of feature words. We thus test 1-, 2- and 3-grams individually, as well as the combination of 1- and 2-grams together (i.e. 1+2-grams) and 1-, 2- and 3-grams (i.e. 1+2+3-grams); this results in five BoW sets. Also, unlike Ivanovic (2008), we test both raw words and lemmas; we expect the use of lemmas to perform better than raw words as our data is less noisy. As the feature weight, in addition to simple Boolean, we also experiment with *TF*, *TF·IDF* and *Information Gain (IG)*.

4.2 Structural Information

Our motivation for using structural information as a feature is that the location of an utterance can be a strong predictor of the dialogue act. That is, dialogues are sequenced, comprising turns (i.e. a given user is sending text), each of which is made up of one or more messages (i.e. strings sent by the user). Structured classification methods which make use of this sequential information have been applied to related tasks such as tagging semantic labels of key sentences in biomedical domains (Chung, 2009) and post labels in web forums (Kim et al., 2010).

Based on the nature of live chats, we observed that the utterance position in the chat, as well as in a turn, plays an important role when identifying its dialogue act. For example, an utterance such as *Hello* will occur at the beginning of a chat while an utterance such as *Have a nice day* will typically appear at the end. The position of utterances in a turn can also help identify the dialogue act; i.e. when there are several utterances in a turn, utterances are related to each other, and thus examining the previous utterances in the same turn can help correctly predict the target utterance. For example, the greeting (*Welcome to ..*) and question (*How may I help you?*) could occur in

Dialogue Act, Definition and Examples
Conventional_Closing: Various ways of ending a conversation e.g. *Bye Bye*
Conventional_Opening: Greeting and other ways of starting a conversation e.g. *Hello Customer*
Downplayer: A backwards-linking label often used after Thanks to down play the contribution e.g. *You are welcome, my pleasure*
Expressive: An acknowledgement of a previous utterance or an indication of the speaker's mood. e.g. *haha,* $:-)$ *wow*
No_Answer: A backward-linking label in the form of a negative response to a YesNo-question e.g. *no, nope*
Open_Question: A question that cannot be answered with only a *yes* or *no*. The answer is usually some form of explanation or statement. e.g. *how do I use the international version?*
Request: Used to express a speaker's desire that the learner do something – either performing some action or simply waiting. e.g. *Please let me know how I can assist you on MSN Shopping today.*
Response_Ack: A backward-linking acknowledgement of the previous utterance. Used to confirm that the previous utterance was received/accepted. e.g. *Sure*
Statement: Used for assertions that may state a belief or commit the speaker to doing something e.g. *I am sending you the page which will pop up in a new window on your screen.*
Thanks: Conventional thanks e.g. *Thank you for contacting us.*
Yes_Answer: A backward-linking label in the form of an affirmative response to a YesNo-Question e.g. *yes, yeah*
YesNo_Question: A closed question which can be answered in the affirmative or negative. e.g. *Did you receive the page, Customer?*

Table 1: The set of dialogue acts used in this research, taken from Ivanovic (2008)

the same turn. We also noticed that identifying the utterance author can help classify the dialogue act (previously used in Ivanovic (2008)).

Based on these observations, we tested the following four structural features:

- Author information,
- Relative position in the chat,
- Author + Relative position,
- Author + Turn-relative position among utterances in a given turn.

We illustrate our structural features in Table 2, which shows an example of a 1-on-1 live chat. The participants are the agent (A) and customer (C); *Uxx* indicates an utterance (U) with ID number xx. This conversation has 42 utterances in total. The relative position is calculated by dividing the utterance number by the total number of utterances in the dialogue; the turn-relative position is calculated by dividing the utterance position by the number of utterances in that turn. For example, for utterance 4 (U4), the relative position is $\frac{4}{42}$, while its turn-relative position is $\frac{2}{3}$ since *U4* is the second utterance among *U3,4,5* that the customer makes in a single turn.

4.3 Utterance Dependency

In recent work, Kim et al. (2010) demonstrated the importance of dependencies between post labels in web forums. The authors introduced series of features based on structural dependencies among posts. They used relative position, author information and automatically predicted labels from previous post(s) as dependency features for assigning a semantic label to the current target post.

Similarly, by examining our chat corpus, we observed significant dependencies between utterances. First, 1-on-1 (i.e. agent-to-user) dialogues often contain dependencies between adjacent utterances by different authors. For example, in Table 2, when the agent asks *Is that correct?*, the expected response from the user is a *Yes* or *No*. Another example is that when the agent makes a greeting, such as *Have a nice day*, then the customer will typically respond with a greeting or closing remark, and not a *Yes* or *No*. Second, the flow of dialogues is in general cohesive, unless the topic of utterances changes dramatically (e.g. *U5: Are you still there?*, *U22: brb in 1 min* in Table 2). Third, we observed that between utterances made by the same author (either agent or user), the target utterance relies on previous utterances made by the same author, especially when

ID	Utterance
A:U1	Hello Customer, welcome to MSN Shopping.
A:U2	My name is Krishna and I am your online Shopping assistant today.
C:U3	Hello!
C:U4	I'm trying to find a sports watch.
C:U5	are you still there?
A:U6	I understand that you are looking for sports watch.
A:U7	Is that correct?
C:U8	yes, that is correct.
..	
C:U22	brb in 1 min
C:U23	Thank you for waiting
..	
A:U37	Thank you for allowing us to assist you regarding wrist watch.
A:U38	I hope you found our session today helpful.
A:U39	If you have any additional questions or you need additional information, please log in again to chat with us. We are available 24 hours a day, 7 days a week for your help.
A:U40	Thank you for contacting MSN Shopping.
A:U41	Have a nice day! Good Bye and Take Care.
C:U42	You too.

Table 2: An example of a 1-on-1 live chat, with turn and utterance structure

the agent and user repeatedly question and answer. With these observations, we checked the likelihood of dialogue act pairings between two adjacent utterances, as well as between two adjacent utterances made by the same author. Overall, we found strong co-occurrence (as measured by number of occurrences of labels across adjacency pairs) between certain pairs of dialogue acts (e.g. (YESNO_QUESTION →YES_ANSWER/NO_ANSWER) and (REQUEST →YES_ANSWER)). STATEMENT, on the other hand, can associate with most other dialogue acts.

Based on this, we designed the following five utterance dependency features; by combining these, we obtain 31 feature sets.

1. Dependency of utterances regardless of author
   (a) Dialogue act of previous utterance
   (b) Accumulated dialogue act(s) of previous utterances
   (c) Accumulated dialogue acts of previous utterances in a given turn
2. Dependency of utterances made by a single author
   (a) Dialogue act of previous utterance by same author; a dialogue act can be in the same turn or in the previous turn
   (b) Accumulated dialogue acts of previous utterances by same author; dialogue acts can be in the same turn or in the previous turn

To capture utterance dependency, Bangalore et al. (2006) previously used n-gram BoW features from the previous 1–3 utterances. In contrast, instead of using utterances which indirectly encode dialogue acts, we directly use the dialogue act classifications, as done in Stolcke et al. (2000). The motivation is that, due to the high performance of simple BoW features, using dialogue acts directly would capture the dependency better than indirect information from utterances, despite introducing some noise. We do not build a probabilistic model of dialogue transitions the way Stolcke et al. (2000) does, but follow an approach similar to that used in Kim et al. (2010) in using predicted dialogue act(s) labels learned in previous step(s) as a feature.

5 Experiment Setup

As stated earlier, we use the data set from Ivanovic (2008) for our experiments; it contains 1-on-1 live chats from an information delivery task. This dataset contains 8 live chats, including 542 manually-segmented utterances. The maximum and minimum number of utterances in a dialogue are 84 and 42, respectively; the maximum number of utterances in a turn is 14. The live chats were manually tagged with the 12 dialogue acts described in Section 3. The utterance distribution over the dialogue acts is described in Table 3.

For our experiments, we calculated TF, TF·IDF and IG (Information Gain) over the utterances, which were optionally lemmatized with the `morph` tool (Minnen et al., 2000). We then built a dialogue act classifier using three different machine learners: SVM-HMM (Joachims, 1998),[2] naive Bayes

[2] http://www.cs.cornell.edu/People/tj/svm_light/svm_hmm.html

Dialogue Act	Utterance number
CONVENTIONAL_CLOSING	15
CONVENTIONAL_OPENING	12
DOWNPLAYER	15
EXPRESSIVE	5
NO_ANSWER	12
OPEN_QUESTION	17
REQUEST	28
RESPONSE _ACK	27
STATEMENT	198
THANKS	79
YES_ANSWER	35
YESNO_QUESTION	99

Table 3: Dialogue act distribution in the corpus

Index	Learner	Ours		Ivanovic	
		Feature	Acc.	Feature	Acc.
Word	SVM	1+2+3/B	.790	1/B	.751
	NB	1/B	.673	1/B	.673
	CRF	1/IG	.839	1/B	.825
Lemma	SVM	1+2+3/IG	.777	N/A	N/A
	NB	1/B	.672	N/A	N/A
	CRF	1/B	**.862**	N/A	N/A

Table 4: Best accuracy achieved by the different learners over different feature sets and weighting methods (1 = 1-gram; 1+2+3 = 1/2/3-grams; B = Boolean; IG = information gain)

from the WEKA machine learning toolkit (Witten and Frank, 2005), and Conditional Random Fields (CRF) using CRF++.[3] Note that we chose to test CRF and SVM-HMM as previous work (e.g. (Samuel et al., 1998; Stolcke et al., 2000; Chung, 2009)) has shown the effectiveness of structured classification models on sequential dependencies. Thus, we expect similar effects with CRF and SVM-HMM. Finally, we ran 8-fold cross-validation using the feature sets described above (partitioning across the 8 sessions). All results are presented in terms of classification accuracy. The accuracy of a zero-R (i.e. majority vote) baseline is 0.36.

6 Evaluation

6.1 Testing Bag-of-Words Features

Table 4 shows the best accuracy achieved by the different learners, in combination with BoW representations and feature weighting methods. Note that the CRF learner ran using 1-grams only, as CRF++ does not accept large numbers of features. As a benchmark, we also tested the method in Ivanovic (2008) and present the best performance over words (rather than lemmas). Overall, we found using just 1-grams produced the best performance for all learners, although SVM achieved the best performance when using all three n-gram orders (i.e. 1+2+3). Since the utterances are very short, 2-grams or 3-grams alone are too sparse to be effective. Among the feature weighting methods, Boolean and IG achieved higher accuracy than TF and TF·IDF. Likewise, due to the short utterances, simple Boolean values were often the most effective. However, as IG was computed using the training data, it also achieved high performance. When comparing the learners, we found that CRF produced the best performance, due to its ability to capture inter-utterance dependencies. Finally, we confirmed that using lemmas results in higher accuracy.

n-gram	Boolean	TF	TF·IDF	IG
1	.731	.511	.517	.766
2	.603	.530	.601	.614
3	.474	.463	.472	.482
1+2	.756	.511	.522	**.777**
1+2+3	.773	.511	.528	**.777**

Table 5: Accuracy of different feature representations and weighting methods for SVM-HMM

Table 5 shows the accuracy over all feature sets; for brevity, we show this for SVM only since the pattern is similar across all learners.

6.2 Using Structural Information

In this section, we describe experiments using structural information—i.e. author and/or position—with BoWs. As with the base BoW technique, we used 1-gram lemmas with Boolean values, based on the results from Section 6.1. Table 6 shows the results: *Pos* indicates the relative position of an utterance in the whole dialogue, *Author* means author information, and Pos_{turn} indicates the relative position of the utterance in a turn. All methods outperformed the baseline; methods that surpassed the results for the simple BoW method (for the given learner) at a

[3] http://crfpp.sourceforge.net/

Feature	Learners		
	CRF	SVM	NB
BoW	.862	.731	.672
BoW+Author	.860	.655	.649
BoW+Pos	.862	.721	.655
BoW+Pos$_{absolute}$	**.863**	.631	.524
BoW+Author+Pos	**.875**	.700	.642
BoW+Author+Pos$_{turn}$	**.871**	.651	.631

Table 6: Accuracy with structural information

level of statistical significance (based on randomised estimation, $p < 0.05$) are boldfaced.

Overall, using CRFs with Author and Position information produced better performance than using BoW alone. Clearly, the ability of CRFs to natively optimise over structural dependencies provides an advantage over other learners.

Relative position cannot of course be measured directly in an actual online application; hence Table 6 also includes the use of "absolute position" as a feature. We see that, for CRF, the absolute position feature shows an insignificant drop in accuracy as compared to the use of relative position. (However, we do see a significant drop in performance when using this feature with SVM and NB.)

6.3 Using Utterance Dependency

We next combined the inter-utterance dependency features with the BoW features. Since we use the dialogue acts directly in utterance dependency, we first experimented using gold-standard dialogue act labels. We also tested using the dialogue acts which were automatically learned in previous steps.

Table 7 shows performance using both the gold-standard and learned dialogue acts. The different features listed are as follows: *LabelList/L* indicates those corresponding to all utterances in a dialogue preceding the target utterance; *LabelPrev/P* indicates a dialogue act from a previous utterance; *LabelAuthor/A* indicates a dialogue act from a previous utterance by the same author; and *LabelPrev*$_t$*/LabelAuthor*$_t$ indicates the previous utterance(s) and previously same-authored utterance(s) in a turn, respectively. Since the accuracy for SVM and NB using learned labels is similar to that using gold standard labels, for brevity we report the performance for CRF using learned labels only. Results that exceed the BoW accuracy at a level of statistical significance ($p < 0.05$) are boldfaced.

Features	Dialogue Acts			
	Goldstandard			**Learned**
	CRF	HMM	NB	CRF
BoW	.862	.731	.672	.862
BoW+LabelList(L)	.795	.435	.225	.803
BoW+LabelPrev(P)	**.875**	.661	.364	**.876**
BoW+LabelAuthor(A)	**.865**	.633	.559	**.865**
BoW+LabelPrev$_t$(P$_t$)	**.873**	.603	.557	**.873**
BoW+LabelAuthor$_t$(A$_t$)	.862	.587	.535	.851
BoW+L+P	.804	.428	.227	.808
BoW+L+A	.799	.404	.225	.804
BoW+L+P$_t$	.803	.413	.229	.804
BoW+L+A$_t$	.808	.408	.216	.801
BoW+P+A	**.873**	.631	.517	**.869**
BoW+P+P$_t$	**.878**	.579	.539	**.875**
BoW+P+A$_t$	**.871**	.603	.519	**.867**
BoW+A+P$_t$	.847	.594	.519	.849
BoW+A+A$_t$	**.869**	.594	.530	**.871**
BoW+P$_t$+A$_t$	**.871**	.592	.519	**.867**
BoW+L+P+A	.812	.419	.231	.804
BoW+L+P+P$_t$	.816	.423	.229	.812
BoW+L+P+A$_t$	.808	.397	.225	.806
BoW+L+A+P$_t$	.810	.388	.225	.810
BoW+L+A+A$_t$	.812	.415	.216	.801
BoW+L+P$_t$+A$_t$	.810	.375	.205	.816
BoW+P+A+P$_t$	**.875**	.602	.522	**.876**
BoW+P+A+A$_t$	.862	.609	.511	**.864**
BoW+P+P$_t$+A$_t$	**.873**	.594	.515	**.867**
BoW+A+P$_t$+A$_t$	**.865**	.594	.517	**.864**
BoW+L+P+A+P$_t$	.817	.410	.231	.810
BoW+L+P+A+A$_t$	.814	.411	.223	.810
BoW+L+P+P$_t$+A$_t$	.816	.382	.205	.806
BoW+L+A+P$_t$+A$_t$	.812	.406	.203	.808
BoW+P+A+P$_t$+A$_t$	**.865**	.583	.513	**.865**
BoW+L+P+A+P$_t$+A$_t$	.816	.399	.205	.803

Table 7: Accuracy for the different learners with dependency features

Utterance dependency features worked well in combination with CRF only. Individually, *Prev* and *Prev*$_t$ (i.e. BoW+P+P$_t$) helped to achieve higher accuracies, and the *Author* feature was also beneficial. However, *List* decreased the performance, as the flow of dialogues can change, and when a larger history of dialogue acts is included, it tends to introduce noise. Comparing use of gold-standard and learned dialogue acts, the reduction in accuracy was not statistically significant, indicating that we can

Feature	CRF	SVM	NB
C+LabelList	.9557	.4613	.2565
C+LabelPrev	.9649	.6365	.5720
C+LabelAuthor	**.9686**	.6310	.5424
C+LabelPrev$_t$	**.9686**	.5738	.5738
C+LabelAuthor$_t$	.9561	.6125	.5332

Table 8: Accuracy with Structural and Dependency Information: *C* means lemmatized Unigram+Position+Author

achieve high performance on dialogue act classification even with interactively-learned dialogue acts. We believe this demonstrates the robustness of the proposed techniques.

Finally, we tested the combination of features from structural and dependency information. That is, we used a base feature (unigrams with Boolean value), relative position, author information, combined with each of the different dependency features – LabelList, LabelPrev, LabelAuthor, LabelPrev$_t$ and LabelAuthor$_t$.

Table 8 shows the performance when using these combinations, for each dependency feature. As we would expect, CRFs performed well with the combined features since CRFs can incorporate the structural and dependency information; the achieved the highest accuracy of 96.86%.

6.4 Error Analysis and Future Work

Finally, we analyzed the errors of the best-performing feature set (i.e. *BoW+Position+Author+LabelAuthor*). In Table 9, we present a confusion matrix of errors, for CONVENTIONAL_CLOSING (Cl), CONVENTIONAL_OPENING (Op), DOWNPLAYER (Dp), EXPRESSIVE (Ex), NO_ANSWER (No), OPEN_QUESTION (Qu), REQUEST (Rq), RESPONSE_ACK (Ack), STATEMENT (St), THANKS (Ta), YES_ANSWER (Yes), and YESNO_QUESTION (YN). Rows indicate the correct dialogue acts and columns indicate misclassified dialogue acts.

Looking over the data, STATEMENT is a common source of misclassification, as it is the majority class in the data. In particularly, a large number of REQUEST and RESPONSE_ACK utterances were tagged as STATEMENT. We did not include punctuation such as question marks in our feature sets; including this would likely improve results further.

In future work, we plan to investigate methods for automatically cleansing the data to remove typos, and taking account of temporal gaps that can sometimes arise in online chats (e.g. in Table 2, there is a time gap between *C:U22 brb in 1 min* and *C:U23 Thank you for waiting*).

7 Conclusion

We have explored an automated approach for classifying dialogue acts in 1-on-1 live chats in the shopping domain, using bag-of-words (BoW), structural information and utterance dependency features. We found that the BoW features perform remarkably well, with slight improvements when using lemmas rather than words. Including structural and inter-utterance dependency information further improved performance. Of the learners we experimented with, CRFs performed best, due to their ability to natively capture sequential dialogue act dependencies.

Acknowledgements

This research was supported in part by funding from Microsoft Research Asia.

References

J.Allen and M.Core. Draft of DAMSL: Dialog Act Markup in Several Layers. The Multiparty Discourse Group. University of Rochester, Rochester, USA. 1997.

A. Anderson, M. Bader, E. Bard, E. Boyle G.M. Doherty, S. Garrod, S. Isard, J. Kowtko, J. McAllister, J. Miller, C. Sotillo, H.S. Thompson, R. and Weinert. The HCRC Map Task Corpus. *Language and Speech.* 1991, 34, pp. 351–366.

J. Ang, Y. Liu and E. Shriberg. Automatic Dialog Act Segmentation and Classification in Multiparty Meetings. *IEEE International Conference on Acoustics, Speech, and Signal Processing*. 2005, pp, 1061–1064.

S. Bangalore, G. Di Fabbrizio and A. Stent. Learning the Structure of Task-Driven Human-Human Dialogs. *Proceedings of the 21st COLING and 44th ACL.* 2006, pp. 201–208.

H. H. Bui. A general model for online probabilistic plan recognition. *IJCAI.* 2003, pp. 1309–1318.

G.Y Chung. Sentence retrieval for abstracts of randomized controlled trials. *BMC Medical Informatics and Decision Making*. 2009, 9(10), pp. 1–13.

F. Debole and F. Sebastiani. Supervised term weighting for automated text categorization. *18th ACM Symposium on Applied Computing*. 2003, pp. 784–788.

	Cl	Op	Dp	Ex	No	Qu	Rq	Ack	St	Ta	Yes	YN
Op	0	0	0	0	0	0	0	0	0	0	0	2
Cl	0	0	0	0	0	0	0	0	1	1	0	0
Dp	0	0	0	0	0	0	0	0	0	0	0	0
Ex	0	0	0	0	0	0	0	0	0	0	0	0
No	0	0	0	0	0	0	0	0	0	0	0	0
Qu	0	0	0	0	0	0	0	0	0	0	0	0
Rq	0	0	0	0	0	0	0	0	3	0	0	0
Ack	0	0	1	0	1	0	0	0	5	0	0	0
St	0	0	0	0	0	0	1	0	0	0	0	0
Ta	1	0	0	0	0	0	0	0	0	0	0	0
Yes	0	0	0	0	0	0	0	0	0	0	0	0
YN	0	1	0	0	0	0	0	0	0	0	0	0

Table 9: Confusion matrix for errors from the CRF with *BoW+Position+Author+LabelAuthor* (rows = correct classification; columns = misclassification; Conventional_Closing = Cl; Conventional_Opening = Op; Downplayer = Dp; Expressive = Ex; No_Answer = No; Open_Question = Qu; Request = Rq; Response_Ack = Ack; Statement = St; Thanks = Ta; Yes_Answer = Yes; and YesNo_Question = YN)

E. N. Forsyth. Improving Automated Lexical and Discourse Analysis of Online Chat Dialog. *Master's thesis*. Naval Postgraduate School, 2007.

J. Godfrey and E. Holliman and J. McDaniel. SWITCHBOARD: Telephone speech corpus for research and development. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*. 1992, pp. 517–520.

S. Grau, E. Sanchis, M. Jose and D. Vilar. Dialogue act classification using a Bayesian approach. *Proceedings of the 9th Conference on Speech and Computer*. 2004.

P. A. Heeman and J. Allen. The Trains 93 Dialogues. *Trains Technical Note 94-2*. Computer Science Dept., University of Rochester, March 1995.

T. Joachims. Text categorization with support vector machines: Learning with many relevant features. *Proceedings of European Conference on Machine Learning*. 1998, pp. 137–142.

M. Johnston, S. Bangalore, G. Vasireddy, A. Stent, P. Ehlen, M. Walker, S. Whittaker and P. Maloor. MATCH: An Architecture for Multimodal Dialogue Systems. *Proceedings of 40th ACL*. 2002, pp. 376–383.

F. N. Julia and K. M. Iftekharuddin. Dialog Act classification using acoustic and discourse information of MapTask Data. *Proceedings of the International Joint Conference on Neural Networks*. 2008, pp. 1472–1479.

D. Jurafsky, E. Shriberg, B Fox and T. Curl. Lexical, Prosodic, and Syntactic Cues for Dialog Acts. *Proceedings of ACL/COLING-98 Workshop on Discourse Relations and Discourse Markers*. 1998, pp. 114–120.

E. Ivanovic. Automatic instant messaging dialogue using statistical models and dialogue acts. *Master's Thesis*. The University of Melbourne. 2008.

S. Keizer. A Bayesian Approach to Dialogue Act Classification. *5th Workshop on Formal Semantics and Pragmatics of Dialogue*. 2001, pp. 210–218.

S.N. Kim and L. Wang and T. Baldwin. Tagging and Linking Web Forum Posts. *Fourteenth Conference on Computational Natural Language Learning*. 2010.

J. Lafferty, A. McCallum and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of ICML*. 2001, pp. 282–289.

D. J. Litman and S. Silliman. ITSPOKE: An Intelligent Tutoring Spoken Dialogue SYstem. *Proceedings of the HLT/NAACL*. 2004.

M. M. Louwerse and S. Crossley. Dialog Act Classification Using N-Gram Algorithms. *FLAIRS Conference*, 2006, pp. 758–763.

G. Minnen, J. Carroll and D. Pearce. Applied morphological processing of English *Natural Language Engineering* 2000, 7(3), pp. 77–80.

M. Purver, J. Niekrasz and S. Peters. Ontology-Based Discourse Understanding for a Persistent Meeting Assistant. *Proc. CHI 2005 Workshop on The Virtuality Continuum Revisited*. 2005.

K. Samuel, Sandra Carberry and K. Vijay-Shanker. Dialogue Act Tagging with Transformation-Based Learning. *Proceedings of COLING/ACL 1998*. 1998, pp. 1150-1156.

E. Shriberg, R. Bates, P. Taylor, A. Stolcke, D. Jurafsky, K. Ries, N. Coccaro, R. Martin, M. Meteer and C. Van

Ess-Dykema. Can Prosody Aid the Automatic Classification of Dialog Acts in Conversational Speech?. *Language and Speech*. 1998, 41(3-4), pp. 439–487.

V. R. Sridhar, S. Bangalore and S. Narayanan. Combining lexical, syntactic and prosodic cues for improved online dialog act tagging. *Computer Speech and Language*. 2009, 23(4), pp. 407–422.

A. Stolcke, K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, P. Taylor, R. Martin, C. Van Ess-Dykema and M. Meteer. Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech. *Computational Linguistics*. 2000, 26(3), pp. 339–373.

A. Stolcke and E. Shriberg. Markovian Combination of Language and Prosodic Models for better Speech Understanding and Recognition . Invited talk at the IEEE Workshop on Speech Recognition and Understanding, Madonna di Campiglio, Italy, December 2001 2001,

C. C. Werry. Linguistic and interactional features of Internet Relay Chat. In S. C. Herring (ed.). *Computer-Mediated Communication*. Benjamins, 1996.

I. Witten and E. Frank. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, 2005.

T. Wu, F. M. Khan, T. A. Fisher, L. A. Shuler and W. M. Pottenger. Posting act tagging using transformation-based learning. *Proceedings of the Workshop on Foundations of Data Mining and Discovery, IEEE International Conference on Data Mining*. 2002.

Resolving Event Noun Phrases to Their Verbal Mentions

Chen Bin[1] **Su Jian**[2] **Tan Chew Lim**[1]

[1]National University of Singapore
{chenbin,tancl}@comp.nus.edu.sg

[2]Institute for Infocomm Research, Singapore
sujian@i2r.a-star.edu.sg

Abstract

Event Anaphora Resolution is an important task for cascaded event template extraction and other NLP study. Previous study only touched on event pronoun resolution. In this paper, we provide the first systematic study to resolve event noun phrases to their verbal mentions crossing long distances. Our study shows various lexical, syntactic and positional features are needed for event noun phrase resolution and most of them, such as morphology relation, synonym and etc, are different from those features used for conventional noun phrase resolution. Syntactic structural information in the parse tree modeled with tree kernel is combined with the above diverse flat features using a composite kernel, which shows more than 10% F-score improvement over the flat features baseline. In addition, we employed a twin-candidate based model to capture the pair-wise candidate preference knowledge, which further demonstrates a statistically significant improvement. All the above contributes to an encouraging performance of 61.36% F-score on OntoNotes corpus.

1 Introduction

Anaphora resolution, the task of resolving a given text expression to its referred expression in prior texts, is important for intelligent text processing systems. Most previous works on anaphora resolution aim at object anaphora which both the anaphor and its antecedent are mentions of the same real world object.

In contrast, an event anaphora[1] as defined in (Asher, 1993) is an anaphoric reference to an event, fact, and proposition which is representative of eventuality and abstract objects. Consider the following example:

There has been ***[the first break in the case]****.*
...
The investigation into the attack which crippled the ship took a positive turn when Yemeni investigators ***[discovered]*** *bomb making equipment in a house said to be close to the port where the ship is anchored.*
...
Now that investigators have had ***[their first major break]****, the confidence level here has risen, but U.S. officials still warn a long investigation lies ahead.*

The two anaphors ***[the first break in the case]*** and ***[their first major break]*** in the above example refer to the same event, "*Yemeni investigators* ***[discovered]*** b*omb making equipment.*" Here, we take the main verb of the event, ***[discovered]*** as the representation of this event and the antecedent of the two anaphors.

Event anaphora (both pronouns and noun phrases) contributes a significant proportion in an anaphora corpus. For example, OntoNotes has 19.97% of its entity chains contains at least one verb mention. Event anaphora resolution also provides critical links for cascaded event template extraction. It provides useful information for the further inference in other natural language processing (NLP) tasks such as discourse relation and entailment as well. Consider the following sentences from OntoNotes,

"In northern Iraq, U.S. warplanes ***[hit]*** *targets including a ridge east of Mosul, where Iraqi troops have been entrenched.*
Two F Tomcats ***[struck]*** *the targets.*
After ***[today's air strikes]****, 13 Iraqi soldiers abandoned their posts and surrendered to Kurdish fighters."*

Resolving the event chain ***[hit] - [struck] - [today's air strikes]*** will provide us details about the "*air strike*" event mentioned in different sentences and

[1] The definition according to (Asher, 1993) includes both gerunds (e.g. destruction) and inflectional verbs (e.g. destroying). In our study, we only focus on the inflectional verbs, as the gerunds are well studied in the conventional anaphora resolution systems.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 872–881,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

also provide us a clue for a temporal/causal relation between these two events, *"Two F Tomcats struck the targets"* and *"13 Iraqi soldiers abandoned their posts and surrendered to Kurdish fighters"*.

In (Asher, 1993) chapter 6, a method to resolve the references to abstract entities using discourse representation theory is discussed. However, no computational system was proposed for event anaphora resolution. (Byron, 2002; Müller, 2007; Chen *et al,* 2010) attempted event pronoun resolution. (Byron, 2002) proposed a knowledge deep approach for a much focused domain like trains spoken dialogue addressed in the paper. Their system resolved limited number of verbs with handcraft knowledge of relevant events. Clearly this approach is not suitable for general event anaphora resolution such as in news articles. Besides, there's also no performance report dedicated on event pronoun resolution, thus it's not clear how effective their approach is. (Müller, 2007) proposed a pronoun resolution system using a set of hand-crafted constraints such as "argumenthood" and "right-frontier condition" together with logistic regression model based on corpus counts. Their system targeted only three pronouns namely, *"it", "this"* and *"that"*. The event pronouns are resolved together with object pronouns. This preliminary explorative work only produced 11.94% F-score for event pronoun resolution which demonstrated the difficulties for event anaphora resolution. In our previous work (Chen *et al,* 2010), we proposed an event pronoun resolution system using various flat and structural knowledge. We managed to achieve a F-score of 57.9% on event pronouns. This paper is a significant improvement and extension from our previous one. Besides, (Pradhan, *et.al,* 2007) applied a conventional co-reference[2] resolution system to OntoNotes corpus using the same set of features for object noun phrase (NP) resolution. There is no specific performance reported on event anaphora resolution. According to our investigation elaborated in section 2.2, the event anaphors may not be correctly resolved in general, as majority of these features are inappropriate for event anaphora resolution.

In this paper, we provide the first systematic study to resolve NPs to event verbs. First, we explore various lexical, positional and syntactic features useful for the event NP resolution, which turns out quite different from conventional anaphora resolution except the sentence distance information. Syntactic structural information is further incorporated using a composite kernel. Furthermore, the candidate preference information is employed using a twin-candidate model. Our approach shows encouraging performance, 61.39% F-score on OntoNotes.

The rest of this paper is organized as follows. Section 2 introduces the framework and various features useful for event NP resolution. Section 3 presents the structural syntactic features and kernel functions to incorporate such features. Twin-candidate model is further introduced to capture the preference knowledge. Section 4 presents the experiment results with discussions. Section 5 concludes the paper.

2 The Resolution Framework

Our event NP resolution system adopts the common learning-based framework for object anaphora/co-reference resolution, as employed by (Soon *et al.*, 2001) and (Ng and Cardie, 2002a).

2.1 Training and Testing instance

In this learning framework, a training/testing instance has a form of $fv(candi_i, ana)$ where $candi_i$ is the i^{th} antecedent candidate of an anaphor ana. An instance is labeled as positive if $candi_i$ is the antecedent of ana, or negative if $candi_i$ is not. An instance is associated with a feature vector which records different properties and relations between ana and $candi_i$. These features will be discussed later in the paper.

During training, for each event NP, we will consider the preceding and succeeding verbs as antecedent candidates. The succeeding verbs are included to accommodate the cataphora phenomenon in which an antecedent occurs after its anaphor. A positive instance is formed by pairing the anaphor with its antecedent. And a set of negative instances is formed by pairing the anaphor with each of its candidates other than the antecedent, which follows the same negative instance selection strategy discussed in (Ng and Cardie, 2002a). Based on these generated training instances, we can train a

[2] Co-reference means two expressions denoting the same entity (e.g. "*admit guilty*" – "*confess*") while anaphora means the latter expression requires the earlier one's information for a correct interpretation (e.g. "*confess*" – "*the confession*"). Despite the difference in definitions, they share a similar set of features in resolution models.

binary classifier using any discriminative learning algorithm.

Testing instances are generated in the same manner except that all the preceding and succeeding verbs will be considered as candidates.

2.2 Flat Features

Table 1 gives a partial list of features used in conventional NP anaphora/co-reference resolution which focuses on objects in (Soon *et al*, 2001; Ng and Cardie, 2002b; Yang *et al*, 2003; Luo *et al*, 2004).

However, most of these features are not useful for our task except the shallow positional features. In event NP resolution, we focus on events instead of objects. Thus the features describing object characteristics such as number agreement, gender agreement and name alias will no longer function here. Secondly, our anaphor and antecedent pair consists of a verb and an NP. The difference in word syntactic categories will introduce extra difficulties using the conventional lexical features such as string matching and head matching. Furthermore, the difference in word syntactic categories will cripple the NP characteristic features for half of the pair. Grammatical features such as appositive structure are no long useful as well.

Conventional Features	*Applicable to Event Anaphora Resolution*
Positional Features	
Sentence Distance	Yes
Object Characteristics	
Number Agreement	No
Gender Agreement	No
Name alias	No
Lexical Features	
String Matching	No
Head Phrase Matching	No
Grammar Features	
Appositive Structure	No
NP Characteristics	
Definite / Indefinite NP	Partial
Demonstrative / Non-Demo NP	Partial
NP is a Proper Name	Partial

Table 1: Features for Conventional NP Resolution

Thus we have conducted a study on the effectiveness of potential important features for event NP resolution. They are elaborated in detail in the rest of this section.

- **Morphological Feature**

Morphological feature captures the inflectional and derivational relation between an anaphor and its antecedent candidate. The morphological feature helps to bridge the gap between different word syntactic categories. This feature represents how close the anaphor and a candidate are in their meanings. A candidate with an inflectional or derivational relation with the anaphor is more preferred to be the antecedent. For example, *"confess"* will be a better antecedent choice for *"confession"* comparing to other verbs. WordNet is used as our morphology knowledge source.

- **Synonym Feature**

Synonym feature is also to capture the similarity in meanings between the anaphor and its antecedent candidates. For example, *"assault"* is a preferable candidate for anaphor *"attack"* (in noun category). In the actual resolution, synonyms are generated from the derivational forms of the anaphor and candidates. This is to overcome the gap in word syntactic categories between an anaphor and its candidates. Two lists of synonyms (including synonyms of derivational forms) are generated for the anaphor and its candidate respectively. The synonym feature will be evaluated by comparing the two lists. Feature values include cases as "**B**oth are **I**n the others' synonym **L**ist (**BIL**)", "**O**ne **I**n the other's **L**ist (**OIL**)", "**L**ists are **O**verlapping (**LO**)" and "**L**ists are **M**utually **E**xclusive (**LME**)". These four values are considered as ordinal with a descending order of **BIL**>**OIL**>**LO**>**LME**. Higher order indicates a more similar word meaning between a candidate and the anaphor. WordNet is used during the synonym lists generation.

- **Fixed Pairings**

Fixed pairings are a list of commonly used referential pairs. For example, *"say - information"* and *"announce - statement"* are commonly used in an anaphoric relation. From a linguistics point of view, *"information"* is the patient role in a *"say"* action. The relation between *"say"* and *"information"* is different from the synonymy and morphology relation described previously. Fixed pairing list is automatically generated from training data by memorizing all encountered pairs of the head of NP anaphor and its verb antecedent. It is 1 if a candidate anaphor pair makes a hit in the fixed pairing list and 0 if the pair does not exist in the pairing list.

- **Named Entity Feature**

This feature indicates if a given NP is a named entity. A named entity is recognized using named entity recognizer for object entities representing person, location, organization and etc. An NP marked as person, location and organization is very unlikely to represent an event. This feature provides a strong heuristic to rule out inappropriate candidates.

- **Contextual Information Features**

This group of features measures the similarities and referential relations exist in the contexts of an anaphor and one of its candidates. These features are derived based on the following two intuitions. First, an event is not only represented by its main verb, the related information (e.g. roles of the action) can be extracted from surrounding contexts. Second, when an event is referred in a later occurrence, the related information is likely to reoccur in the contexts as well. Therefore, this group of feature is designed to capture such knowledge. There are two features in this group.

Context Words Similarity

This feature measures similarity between anaphor's contexts and its candidate's context. Stop words (such as *"in", "the"* and etc.) are removed from the contexts before calculating the similarity. The similarity is calculated based on a list of nearby 10 contextual words. The number of common words is used to represent the contextual similarity. Inflectional and derivational forms in the contextual words are considered as match cases.

Co-referential Relation(s) in Contexts

This feature is 1 if at least one object co-referential relation exists between the anaphor's contexts and its candidate's contexts. The idea is still to capture matching roles of action in the two contexts. For example,

> *"**[George W. Bush]**$_1$ **{approved}**$_2$ the new military plan .... **{[The president]**$_1$ **'s decision}**$_2$ agitated various anti-war groups ...".*

By knowing that the ***[George W. Bush]***$_1$ and ***[The president]***$_1$ co-refer with each other, ***{approved}***$_2$ is a preferable candidate for ***{The president's decision}***$_2$ as they share a common attribute value "*president Bush*".

- **NP Antecedent(s) Features**

When an NP co-refers a preceding NP, the original phrase will normally be replaced with a more concise expression which is the anaphor. By using the full expression from the antecedent, we can obtain extra knowledge for the later concise expression. For the antecedent knowledge of NPs, we used the OntoNote gold standard annotations for object co-references. There are 3 features in this group.

Morphological Feature with NP's Antecedent(s)

This feature is evaluated by comparing each of an NP's antecedents with its verb candidates for an inflectional or derivational relation. It is considered as a morphological relation if one of the NP's antecedents is inflectional or derivational to the verb.

Synonym Feature with NP's Antecedent(s)

Similar to the above, the synonym list generated from an NP's co-referential expressions is used to compare with its verb candidate's synonym list. The final feature value is taken to be the highest order as described in the previous section on synonym feature.

Named Entity Feature with NP's Antecedent(s)

Similar to the NP's named entity feature described previously, this feature is used to rule out inappropriate NPs for an event anaphoric relation. Consider the object co-referential expressions *"George W. Bush"* and *"the president"*, the first one will be marked as named entity but not the latter. By using the object NP's co-reference knowledge, we can rule out the inappropriate NP *"the president"* as it refers to an object.

- **Grammatical Role**

This set of feature aims to capture the grammatical roles of the anaphor and its antecedent candidates. The details of this set of features are tabulated in Table 2 below.

NP: *M*	
Sbj_Main	1 if *M* is subject in main clause; else 0.
Sbj_Sub	1 if *M* is subject in sub-clause; else 0.
Obj_Main	1 if *M* is object in main clause; else 0.
Obj_Sub	1 if *M* is object in sub-clause; else 0.
Verb: *V*	
Main	1 if *V* in main clause; else 0.
Sub	1 if *V* in sub-clause; else 0.

Table 2: Features representing Grammatical Roles

- **Positional and NP Characteristics Features**

Positional Features:	***NP:** M**; Verb:** V*
SentDist	# of Sentences between *M* and *V*;
PhraseDist	# of NPs between *M* and *V;*
WordDist	# of words between *M* and *V;*
NP Characteristic Features:	
NP_Def	1 if *M* is definite; else 0;
NP_Demo	1 if *M* is demonstrative; else 0;
NP_First	1 if *M* is the first NP in its sentence;

Table 3: Positional & NP Characteristic Features

A set of positional and NP characteristic features is employed in our resolution system. Positional features extend their counterparts in conventional NP anaphora resolution by incorporating phrase distance and word distance. NP characteristic features are applied to (possible) NP Anaphora. These features are tabulated in Table 3.

3 Incorporating Structural Syntactic Information

A parse tree that covers an event NP and its antecedent could provide us much syntactic information related to the pair. The commonly used syntactic knowledge for anaphora resolution, such as the governing relations, can be directly described by the tree structure. Other syntactic knowledge that may be helpful for anaphora resolution could also be implicitly represented in the parse tree. Such syntactic knowledge can be captured using a convolution tree kernel by comparing the number of common sub-structures in two trees. The implicit syntactic structural knowledge can be further combined with other knowledge through a composite kernel.

Normally, parsing is done on the sentence level. However, in many cases an event NP and its antecedent do not occur in the same sentence. To present their syntactic properties and relations in a single tree structure, we construct a syntax tree for the entire text, by attaching the parse trees of all its sentences to a pseudo upper node. Having obtained the parse tree of a text, we shall consider how to select the appropriate portion of the tree as the structural feature for a given instance. As each instance is related to an event NP and one of its candidates, the structured feature at least should be able to cover both of these two expressions.

3.1 Structural Syntactic Feature

Generally, the more portion of the parse tree is included, the more syntactic information would be provided. But at the same time, the more noisy information that comes from parsing errors and other sources would likely be introduced as well. In our study, we examine three possible structural features that contain different portions of the parse tree:

- **Minimum Expansion Tree**

This feature records the minimal structure covering both the NP and its verb candidate in the parse tree. It only includes the nodes occurring in the shortest path connecting the NP and its candidate, via the nearest commonly commanding node. When the anaphor and its antecedent are from different sentences, we will find a path through a pseudo “TOP” node which links all the parse trees of sentences of a text.

- **Simple Expansion Tree**

Minimum-Expansion could, to some degree, describe the syntactic relationships between an anaphor and its candidate. However, the tree structure surrounding the expression is not taken into consideration. To incorporate such information, feature Simple-Expansion not only contains all the nodes in Minimum-Expansion, but also includes the first-level children of those nodes between the antecedent and anaphor pair except the punctuations. Thus, Simple-Expansion contains a concise representation of surrounding syntax structures.

- **Full Expansion Tree**

This feature focuses on the whole tree structure between the candidate and anaphor. It not only includes all the nodes in Simple-Expansion, but also the nodes (beneath the nearest commanding parent) that cover the words between one candidate and the anaphor. In multi-sentence cases, only sentences containing the anaphors/antecedents are expanded. Such a feature keeps the most information related to the anaphor and candidate pair.

Figure 1 illustrates the differences of three syntactic tree structures. In each expansion tree, the involved sub-tree is shaded in grey with bold fonts and thicker lines.

3.2 Convolution Parse Tree Kernel and Composite Kernel

To capture the structural features in parse tree, we use the convolution tree kernel which is defined in (Collins and Duffy, 2002) and (Moschitti, 2004). Given two trees, the kernel will enumerate all their sub-trees and compute the number of common subtrees. As been proved, the convolution kernel can be efficiently computed in polynomial time on average. The above tree kernel only aims for the structured features. We also need a composite kernel to combine together the structured features and the flat features described in section 2.2. In our study we define the composite kernel as follows:

$$K_{comp}(x_1, x_2) = \frac{K_{tree}(x_1, x_2)}{|K_{tree}(x_1, x_2)|} + \frac{K_{flat}(x_1, x_2)}{|K_{flat}(x_1, x_2)|}$$

where K_{tree} is the convolution tree kernel defined for the structured features, and K_{flat} is the kernel applied to the flat features. Both kernels are divided by their respective length ($|K(x_1, x_2)| =$

$\sqrt{K(x_1,x_1)\cdot K(x_2,x_2)}$) for normalization. The new composite kernel K_{comp}, defined as the summation of normalized K_{tree} and K_{flat}.

3.3 Twin-Candidate Framework using Ranking SVM Model

In a ranking SVM kernel as in (Moschitti *et al,* 2006) for Semantic Role Labeling, two argument annotations (as argument trees) are presented to a ranking SVM model to decide which one is better. In our case, we have a slightly different setting. Instead of two argument trees, we present two syntactic trees from two candidates to the ranking SVM model. The idea is inspired by (Yang, *et.al,* 2005;2008). The intuition behind the twin-candidate model is to capture the information of how much one candidate is more preferred than another. The candidate wins most of the pair-wise comparisons will be selected as the antecedent.

The feature vector for each training instance has a form of $fv = (candi_i, candi_j)$. An instance is positive if $candi_i$ is a better choice than $candi_j$. Otherwise, it is a negative instance. For each feature vector, both tree structural features and flat features are used. Thus each feature vector has a detailed form of $fv = (t_i, t_j, v_i, v_j)$ where t_i and t_j are the parse trees of candidate i and j respectively; v_i and v_j are the flat feature vectors of candidate i and j respectively. Therefore the final SVM kernel is a joint kernel of tree kernel and flat kernel. In the training instances generation, we only generate those instances with one candidate being the antecedent. This follows the same strategy used in (Yang *et al*, 2008) for object anaphora resolution.

In the resolution process, a list of m candidates is extracted as described in section 2.1. A total of $\binom{m}{2}$ instances are generated by pairing-up the m candidates pair-wisely. We used a Round-Robin scoring scheme for antecedent selection. Suppose a SVM output for a testing instance $fv = (candi_i, candi_j)$ is positive, we will give a score 1 for $candi_i$ and -1 for $candi_j$. Similarly, if a SVM output for a testing instance $fv = (candi_i, candi_j)$ is negative, we will give a score -1 for $candi_i$ and 1 for $candi_j$. At last, the candidate with the highest final score is selected as the antecedent[3]. In order to handle a non-event NP encountered during the resolution, we empirically set a threshold to distinguish event anaphoric from non-event anaphoric.

4 Experiments and Discussions

4.1 Experimental Setup

OntoNotes Release 2.0 English corpus is used in our study. It contains 300k words of English newswire data (from the Wall Street Journal) and 200k words of English broadcast news data (from ABC, CNN, NBC, Public Radio International and Voice of America). Table 4 shows nearly 20% of the entity chains annotated in OntoNotes are event chains with at least one verb mention. This significant proportion demonstrates importance of event anaphora resolution in a text processing system.

Chain Type	*Count*	*Percentage*
Event Chain	1235	19.97%
Object Chain	4952	80.03%
Total	6187	100%

Table 4: Percentage of Event Chains in OntoNotes

Our resolution system focuses on the verb-NP links existing in the event chains. In total, we have extracted 977 verb-NP pairs from the OntoNotes corpus. To illustrate the task difficulties, the sentence distance between the anaphor and its antecedent is tabulated below in Table 5. The average sentence distance is 2.97.

SenDist	**0**	**1**	**2**	**3**	**4**	**5**
Count	102	436	143	66	38	69
SenDist	**6**	**7**	**8**	**9**	**10**	**>10**
Count	14	15	11	13	7	63

Table 5: Distribution of Sentence Distance

Due to the long separation distance, the number of candidates during resolution is very large. There are on average 30.49 candidates in training and 583.74 candidates in testing. There is no simple baseline for the task. The most recent verb for each NP will result an F-score almost zero as majority (78.7%) of the NPs are not anaphoric in OntoNotes**.** On the other hand, only 2.31% of verbs in OntoNotes are referred by an NP.

To conduct event NP resolution, an input raw text was preprocessed automatically by a pipeline of NLP components. The NP identification and the predicate-argument extraction were done based on Stanford Parser (Klein and Manning, 2003a;b) with F-score of 86.32% on Penn Treebank corpus. The named entity recognition process uses a SVM based NER trained and tested on ACE 2005 with 92.5(P) 84.3(R) 88.2(F).

[3] In a tie-breaking scenario, the candidate closer to anaphor is selected as antecedent.

4.2 Experiment Results and Discussion

In this section, we will present our experimental results with discussions. The performance measures we used are precision, recall and F-score. All the experiments are done with a 10-folds cross validation to evaluate the performances. In each experiment, the whole corpus is divided into 10 equal sized portions. In each fold, one of the portions is selected to be testing corpus while the remaining 9 are used for training. In case of statistical significance test for differences is needed, a one-tailed, paired-sample Student's t-Test is performed at 0.05 level of significance.

In the first set of experiment results, we are investigating effectiveness of each flat feature. The effectiveness of an individual feature is measured in a leave-one-out manner. That is the performance loss by removing a particular feature from the feature list. The greater performance drop after removing a feature indicates the more effective that feature is for event NP resolution. Table 6 presents the results of this set of experiments.

Feature	Precision	Recall	F-score
ALL	43.87%	42.86%	43.35%
-Morph	8.74%	5.84%	6.99%
-Synonym	7.24%	4.63%	5.64%
-Fixed_Pair	9.94%	5.43%	7.01%
-NE	12.40%	7.04%	8.95%
-Cont_Sim	10.35%	4.63%	6.37%
-Cont_Coref	8.17%	4.43%	5.72%
-Ante_Morph	11.00%	6.64%	8.26%
-Ante_Syn	11.95%	7.04%	8.84%
-Ante_NE	10.36%	7.24%	8.51%
-Gram_Role	11.76%	6.64%	8.45%
-Position	47.47%	32.11%	38.31%
-NP_Def	11.85%	6.04%	7.99%
-NP_Demo	7.85%	4.23%	5.48%
-NP_First	12.98%	7.04%	9.12%

Table 6: Effectiveness of Individual Flat Feature

In Table 6, the first line is performance using all the flat features. Each line below is the performance after removing the feature in that line from the resolution system. The observations in Table 6 suggest that all the features we have discussed in section 2.2 contribute a significant part in the resolution system. For most of the features (except position), the overall system is almost not functioning for the identification of the antecedent. The performance drops for most of features are over 30% in F-score. The conclusion we can draw from these observations is that the flat features are co-dependent to perform the event NP resolution task. Each feature's individual contribution is hard to be separated from the overall performance. All of them are essential parts in the resolution system. Positional feature will incur a 5.04% drop in F-score. Although it is comparatively smaller than the performance drop of the other features, it is still a significant part in the overall performance. Especially, after removing positional features, the recall decreases by 10.75%. Therefore, in the later on experiments, all the flat features are used for event NP resolution.

In the next set of experiments, we are aiming to investigate the individual effectiveness of each knowledge source. Table 7 reports the performance of these experiments.

	Precision	Recall	F-score
Flat	43.87%	42.86%	43.35%
Min-Exp	33.35%	19.95%	24.82%
Simple-Exp	22.22%	8.45%	12.24%
Full-Exp	33.33%	5.63%	9.63%

Table 7: Contribution from Single Knowledge Source

From Table 7, the flat feature set yields a baseline system with 43.35% F-score. By using each tree structure alone, we can only achieve a performance of 24.82% F-score using the minimum-expansion tree. These results indicate that the syntactic structural information alone cannot resolve event anaphoric noun phrases.

As we explained in section 3.2, a composite kernel can be used to combine flat features with syntactic structure feature. The third set of experiments is conducted to verify the performances of various tree structures combined with the flat features. The performances are reported in Table 8.

	Precision	Recall	F-score
Flat	43.87%	42.86%	43.35%
Flat+Min-Exp	65.78%	53.60%	59.01%
Flat+Sim-Exp	62.85%	49.64%	55.43%
Flat+Full-Exp	64.56%	50.77%	56.77%

Table 8: Comparison of Different Combinations

As Table 8 presents, all the three types of structural information improve the overall performance by over 10% in F-score. Obviously, syntactic structural information is very useful for event NP resolution when combined with flat features. Minimum expansion tree performs better than the other two structures. The performance difference in simple expansion and full expansion are statistically insignificant. This result shows that contextual structural information is considered as harmful rather than helpful in an event NP resolution. The minimum structural information covering the anaphor and antecedent is the most helpful as it introduces least noises. This finding is different from the conclusion in conventional pronoun resolution as reported in (Yang *et al,* 2006;) where simple expansion tree performs best. We consider this difference is caused by the distance of separation from the anaphor to its antecedent.

In the last set of experiment, we will present the performance from the twin-candidates based approach in Table 9. The first line is the best performance from single candidate model. The second line is performance using the twin-candidate approach.

Flat+Min-Exp	**Precision**	**Recall**	**F-score**
Single Candidate	65.78%	53.60%	59.01%
Twin-Candidates	66.41%	57.93%	61.36%

Table 9: Single Candidate V.S. Twin Candidates

Comparing to the single candidate model, the recall is significantly improved with similar level of precision. The difference in results is statistically significant. It reinforced our intuition that preference knowledge between two candidates is a contributive information source in event NP resolution.

Last but not least, we have conducted a study on the performance impact of various training data size. 10% from the whole corpus is kept as testing data. The remaining training data is further split into 10 equal sized portions. The experiments are conducted by gradually increase the number of training portion by one at each run. Due to time constraints, the learning curve experiments are conducted using single candidate model as the twin-candidate model is very time consuming. But it should be representative for twin-candidate scenario as well. The resulted learning curve is plotted in Figure 2.

As presented in Figure 2, the training data size increases, the performance curve quickly converged around 0.55 for F-score when the percentage of training data reaches near 50% of training data. After that, F-score still increases as more training data are used. But the rate of increment slows down. It reaches a relatively flat shape near 80% of training data are used. The last 3 experiment results (corresponding to last 3 points plotted in learning curve) are tested for statistical differences. The results show that they are statistically indifferent which suggests a saturated performance is achieved when the training data proportion reaches 80%.

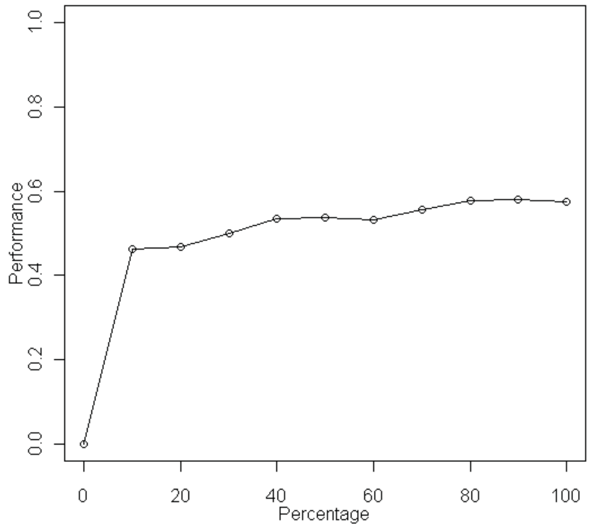


Figure 2: Learning Curve on Various Training Size

This learning curve shows that a satisfactory performance can be achieved with reasonable sized training corpus such as OntoNotes.

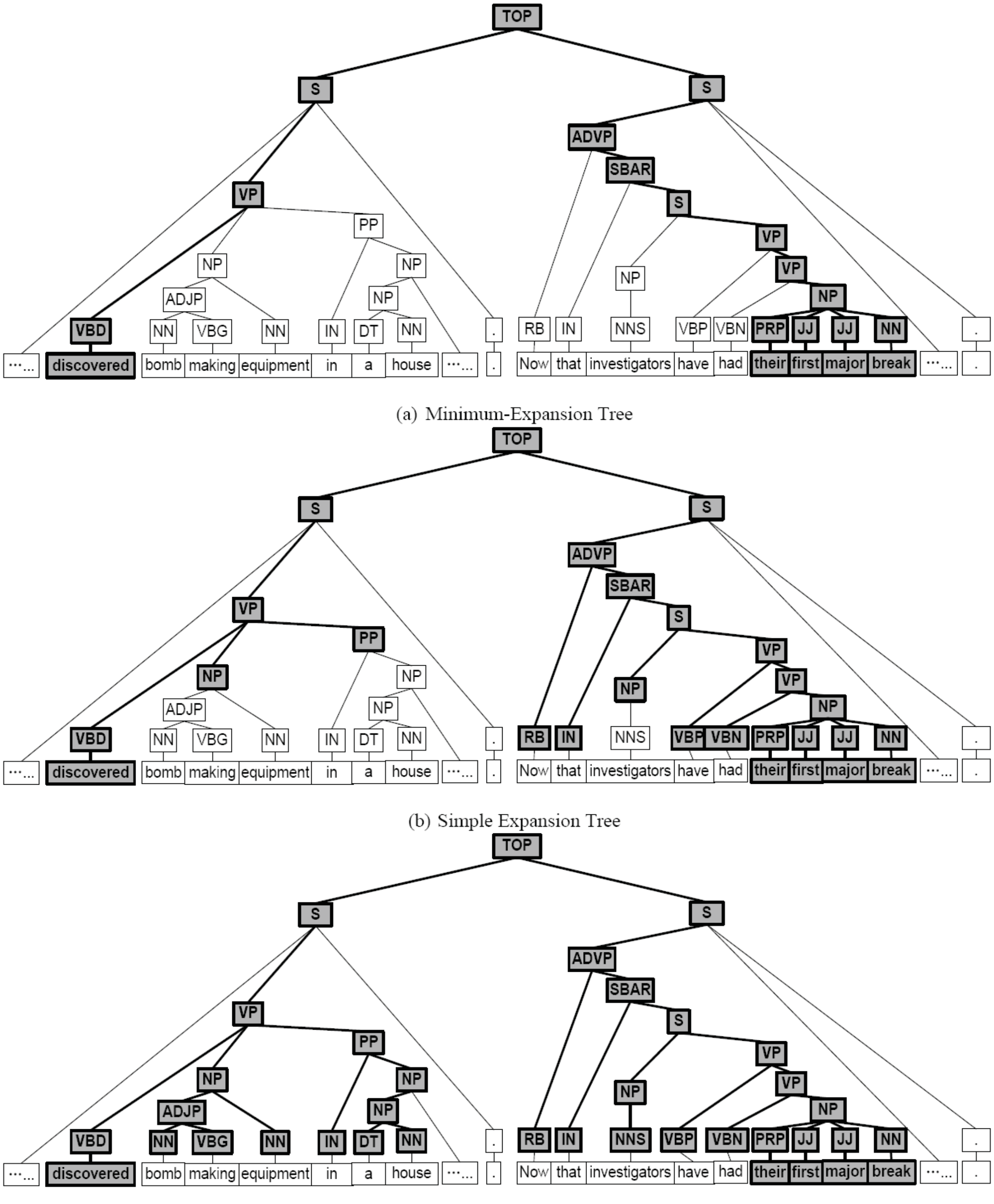
5 Conclusion and Future Work

The purpose of this paper is to conduct a systematic study of the event NP resolution, which is not available in the literature. We propose a resolution system utilizing a set of flat positional, lexical, syntactic features and structural syntactic features. The state-of-arts convolution tree kernel is used to extract indicative structural syntactic knowledge. A twin-candidates preference based approach is incorporated to reinforce the resolution system with candidate preference knowledge. Last but not least, we also proposed a study to examine how various training corpus sizes will affect the resolution performance.

In the future, we would like to further explore more semantic information can be employed into the system such as semantic role labels and verb frames.

Acknowledgement

We would like to thank Professor Massimo Poesio from University of Trento for the initial discussion of this work.



(a) Minimum-Expansion Tree

(b) Simple Expansion Tree

(c) Full-Expansion Tree

Figure 1: Comparison Different Syntactic Structures using part of example from section 1.
"… ***[discovered]***$_2$ bomb equipment in a house… .
Now that investigators have had ***[their first major break]***$_3$… ."

References

N. Asher. 1993. *Reference to Abstract Objects in Discourse*. Kluwer Academic Publisher.

V. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer.

T. Joachims. 1999. Making large-scale svm learning practical. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

W. Soon, H. Ng, and D. Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521– 544.

D. Byron. 2002. Resolving Pronominal Reference to Abstract Entities, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, 2002, USA

M. Collins and N. Duffy. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*. July 2002. , USA

V. Ng and C. Cardie. 2002a. Combining sample selection and error-driven pruning for machine learning of coreference rules. In *Proceedings of the conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 55–62, Philadelphia.

V. Ng and C. Cardie. 2002b. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, pages 104–111, Philadelphia.

D. Klein and C. Manning. 2003a. Fast Exact Inference with a Factored Model for Natural Language Parsing. In *Advances in Neural Information Processing Systems 15 (NIPS 2002),* Cambridge, MA: MIT Press, pp. 3-10.

D. Klein and C. Manning. 2003b. Accurate Unlexicalized Parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL'03)*, pp. 423-430.

X. Yang, G. Zhou, J. Su, and C. Tan. 2003. Coreference Resolution Using Competition Learning Approach. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL'03)*, pp 176–183.

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'04)*,2004

A. Moschitti. 2004. A study on convolution kernels for shallow semantic parsing. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL'04)*, pg 335–342.

X. Yang, J. Su and C.Tan. 2005. A Twin-Candidates Model for Coreference Resolution with Non-Anaphoric Identification Capability. In *Proceedings of IJCNLP-2005*. Pp. 719-730, 2005

A. Moschitti, Making tree kernels practical for natural language learning. In *Proceedings EACL 2006*, Trento, Italy, 2006.

X. Yang, J. Su and C.Tan. 2006. Kernel-Based Pronoun Resolution with Structured Syntactic Knowledge. *In Proceedings of ACL 2006*. July 2006. Sydney, Australia.

E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel. 2006. OntoNotes: The 90\% Solution. In *Proceedings of the Human Language Technology Conference of the NAACL*, 2006

S. Pradhan, L. Ramshaw, R. Weischedel, J. MacBride, and L. Micciulla. 2007. Unrestricted Coreference: Identifying Entities and Events in OntoNotes. In *Proceedings of the IEEE International Conference on Semantic Computing (ICSC),* Irvine, CA, Sep. 17-19, 2007.

C. Müller. 2007. Resolving it, this, and that in unrestricted multi-party dialog. *In Proceedings of ACL-07 conference*, pages 816–823.

X. Yang, J. Su and C.Tan. 2008. A Twin-Candidates Model for Learning-Based Coreference Resolution. In *Computational Linguistics*, 34(3):327-356.

G. Miller. 2009. "WordNet–About Us." *WordNet.* Princeton University, 2009 http://wordnet.princeton.edu

B. Chen, J. Su and C. Tan, 2010. A Twin-Candidate Based Approach for Event Pronoun Resolution using Composite Kernel. In *Proceedings of the 23rd International Conference on Computational Linguistics (CoLing'10),* 2010.pg188-196, Beijing, China,

A Tree Kernel-based Unified Framework for Chinese Zero Anaphora Resolution

Fang Kong Guodong Zhou[*]
JiangSu Provincial Key Lab for Computer Information Processing Technology
School of Computer Science and Technology Soochow University
{kongfang, gdzhou}@suda.edu.cn

Abstract

This paper proposes a unified framework for zero anaphora resolution, which can be divided into three sub-tasks: zero anaphor detection, anaphoricity determination and antecedent identification. In particular, all the three sub-tasks are addressed using tree kernel-based methods with appropriate syntactic parse tree structures. Experimental results on a Chinese zero anaphora corpus show that the proposed tree kernel-based methods significantly outperform the feature-based ones. This indicates the critical role of the structural information in zero anaphora resolution and the necessity of tree kernel-based methods in modeling such structural information. To our best knowledge, this is the first systematic work dealing with all the three sub-tasks in Chinese zero anaphora resolution via a unified framework. Moreover, we release a Chinese zero anaphora corpus of 100 documents, which adds a layer of annotation to the manually-parsed sentences in the Chinese Treebank (CTB) 6.0.

1 Introduction

As one of the most important techniques in discourse analysis, anaphora resolution has been a focus of research in Natural Language Processing (NLP) for decades and achieved much success in English recently (e.g. Soon et al. 2001; Ng and Cardie 2002; Yang et al. 2003, 2008; Kong et al. 2009).

However, there is little work on anaphora resolution in Chinese. A major reason for this phenomenon is that Chinese, unlike English, is a pro-drop language, whereas in English, definite noun phrases (e.g. *the company*) and overt pronouns (e.g. *he*) are frequently employed as referring expressions, which refer to preceding entities. Kim (2000) compared the use of overt subjects in English and Chinese. He found that overt subjects occupy over 96% in English, while this percentage drops to only 64% in Chinese. This indicates the prevalence of zero anaphors in Chinese and the necessity of zero anaphora resolution in Chinese anaphora resolution. Since zero anaphors give little hints (e.g. number or gender) about their possible antecedents, zero anaphora resolution is much more challenging than traditional anaphora resolution.

Although Chinese zero anaphora has been widely studied in the linguistics research (Li and Thompson 1979; Li 2004), only a small body of prior work in computational linguistics deals with Chinese zero anaphora resolution (Converse 2006; Zhao and Ng 2007). Moreover, zero anaphor detection, as a critical component for real applications of zero anaphora resolution, has been largely ignored.

This paper proposes a unified framework for Chinese zero anaphora resolution, which can be divided into three sub-tasks: zero anaphor detection, which detects zero anaphors from a text, anaphoricity determination, which determines whether a zero anaphor is anaphoric or not, and antecedent identification, which finds the antecedent for an anaphoric zero anaphor. To our best knowledge, this is the first systematic work dealing with all the three sub-tasks via a unified framework. Moreover, we release a Chinese zero anaphora corpus of 100 documents, which adds a layer of annotation to the

[*] Corresponding author

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 882–891,
MIT, Massachusetts, USA, 9-11 October 2010.

manually-parsed sentences in the Chinese Treebank (CTB) 6.0. This is done by assigning anaphoric/non-anaphoric zero anaphora labels to the null constituents in a parse tree. Finally, this paper illustrates the critical role of the structural information in zero anaphora resolution and the necessity of tree kernel-based methods in modeling such structural information.

The rest of this paper is organized as follows. Section 2 briefly describes the related work on both zero anaphora resolution and tree kernel-based anaphora resolution. Section 3 introduces the overwhelming problem of zero anaphora in Chinese and our developed Chinese zero anaphora corpus, which is available for research purpose. Section 4 presents our tree kernel-based unified framework in zero anaphora resolution. Section 5 reports the experimental results. Finally, we conclude our work in Section 6.

2 Related Work

This section briefly overviews the related work on both zero anaphora resolution and tree kernel-based anaphora resolution.

2.1 Zero anaphora resolution

Although zero anaphors are prevalent in many languages, such as Chinese, Japanese and Spanish, there only have a few works on zero anaphora resolution.

Zero anaphora resolution in Chinese

Converse (2006) developed a Chinese zero anaphora corpus which only deals with zero anaphora category "-NONE- *pro*" for dropped subjects/objects and ignores other categories, such as "-NONE- *PRO*" for non-overt subjects in non-finite clauses. Besides, Converse (2006) proposed a rule-based method to resolve the anaphoric zero anaphors only. The method did not consider zero anaphor detection and anaphoric identification, and performed zero anaphora resolution using the Hobbs algorithm (Hobbs, 1978), assuming the availability of golden anaphoric zero anaphors and golden parse trees.

Instead, Zhao and Ng (2007) proposed feature-based methods to zero anaphora resolution on the same corpus from Convese (2006). However, they only considered zero anaphors with explicit noun phrase referents and discarded those with split antecedents or referring to events. Moreover, they focused on the sub-tasks of anaphoricity determination and antecedent identification. For zero anaphor detection, a simple heuristic rule was employed. Although this rule can recover almost all the zero anaphors, it suffers from very low precision by introducing too many false zero anaphors and thus leads to low performance in anaphoricity determination, much due to the imbalance between positive and negative training examples.

Zero anaphora resolution in Japanese

Seki et al. (2002) proposed a probabilistic model for the sub-tasks of anaphoric identification and antecedent identification with the help of a verb dictionary. They did not perform zero anaphor detection, assuming the availability of golden zero anaphors. Besides, their model needed a large-scale corpus to estimate the probabilities to prevent them from the data sparseness problem.

Isozaki and Hirao (2003) explored some ranking rules and a machine learning method on zero anaphora resolution. However, they assumed that zero anaphors were already detected and each zero anaphor's grammatical case was already determined by a zero anaphor detector.

Iida et al. (2006) explored a machine learning method for the sub-task of antecedent identification using rich syntactic pattern features, assuming the availability of golden anaphoric zero anaphors.

Sasano et al. (2008) proposed a fully-lexicalized probabilistic model for zero anaphora resolution, which estimated case assignments for the overt case components and the antecedents of zero anaphors simultaneously. However, this model needed case frames to detect zero anaphors and a large-scale corpus to construct these case frames automatically.

For Japanese zero anaphora, we do not see any reports about zero anaphora categories. Moreover, all the above related works we can find on Japanese zero anaphora resolution ignore zero anaphor detection, focusing on either anaphoricity determination or antecedent identification. Maybe, it is easy to detect zero anaphors in Japanese. However, it is out of the scope of our knowledge and this paper.

Zero anaphora resolution in Spanish

As the only work we can find, Ferrandez and Peral (2000) proposed a hand-engineered rule-based method for both anaphoricity determination and antecedent identification. That is, they ignored zero anaphor detection. Besides, they only dealt with zero anaphors that were in the subject position.

2.2 Tree kernel-based anaphora resolution

Although there is no research on tree kernel-based zero anaphora resolution in the literature, tree kernel-based methods have been explored in traditional anaphora resolution to certain extent and achieved comparable performance with the dominated feature-based ones. One main advantage of kernel-based methods is that they are very effective at reducing the burden of feature engineering for structured objects. Indeed, the kernel-based methods have been successfully applied to mine structural information in various NLP techniques and applications, such as syntactic parsing (Collins and Duffy 2001; Moschitti 2004), semantic relation extraction (Zelenko et al. 2003; Zhao and Grishman 2005; Zhou et al. 2007; Qian et al. 2008), and semantic role labeling (Moschitti 2004).

Representative works in tree kernel-based anaphora resolution include Yang et al. (2006) and Zhou et al (2008). Yang et al. (2006) employed a convolution tree kernel on anaphora resolution of pronouns. In particular, a document-level syntactic parse tree for an entire text was constructed by attaching the parse trees of all its sentences to a new-added upper node. Examination of three parse tree structures using different construction schemes (Min-Expansion, Simple-Expansion and Full-Expansion) on the ACE 2003 corpus showed promising results. However, among the three constructed parse tree structures, there exists no obvious overwhelming one, which can well cover structured syntactic information. One problem with this tree kernel-based method is that all the constructed parse tree structures are context-free and do not consider the information outside the sub-trees. To overcome this problem, Zhou et al. (2008) proposed a dynamic-expansion scheme to automatically construct a proper parse tree structure for anaphora resolution of pronouns by taking predicate- and antecedent competitor-related information into consideration. Besides, they proposed a context-sensitive convolution tree kernel to compute the similarity between the parse tree structures. Evaluation on the ACE 2003 corpus showed that the dynamic-expansion scheme can well cover necessary structural information in the parse tree for anaphora resolution of pronouns and the context-sensitive convolution tree kernel much outperformed other tree kernels.

3 Task Definition

This section introduces the phenomenon of zero anaphora in Chinese and our developed Chinese zero anaphora corpus.

3.1 Zero anaphora in Chinese

A zero anaphor is a gap in a sentence, which refers to an entity that supplies the necessary information for interpreting the gap. Figure 1 illustrates an example sentence from Chinese TreeBank (CTB) 6.0 (File ID=001, Sentence ID=8). In this example, there are four zero anaphors denoted as Φi (i=1, 2, …4). Generally, zero anaphors can be understood from the context and do not need to be specified.

A zero anaphor can be classified into either anaphoric or non-anaphoric, depending on whether it has an antecedent in the discourse. Typically, a zero anaphor is non-anaphoric when it refers to an extra linguistic entity (e.g. the first or second person in a conversion) or its referent is unspecified in the context. Among the four anaphors in Figure 1, zero anaphors Φ1 and Φ4 are non-anaphoric while zero anaphors Φ2 and Φ3 are anaphoric, referring to noun phrase "建筑行为/building action" and noun phrase "新区管委会/new district managing committee" respectively.

Chinese zero anaphora resolution is very difficult due to following reasons: 1) Zero anaphors give little hints (e.g. number or gender) about their possible antecedents. This makes antecedent identification much more difficult than traditional anaphora resolution. 2) A zero anaphor can be either anaphoric or non-anaphoric. In our corpus described in Section 3.2, about 60% of zero anaphors are non-anaphoric. This indicates the importance of anaphoricity determination. 3) Zero anaphors are not explicitly marked in a text. This indicates the necessity of zero anaphor detection, which has been largely ignored in previous research and has proved to be difficult in our later experiments.

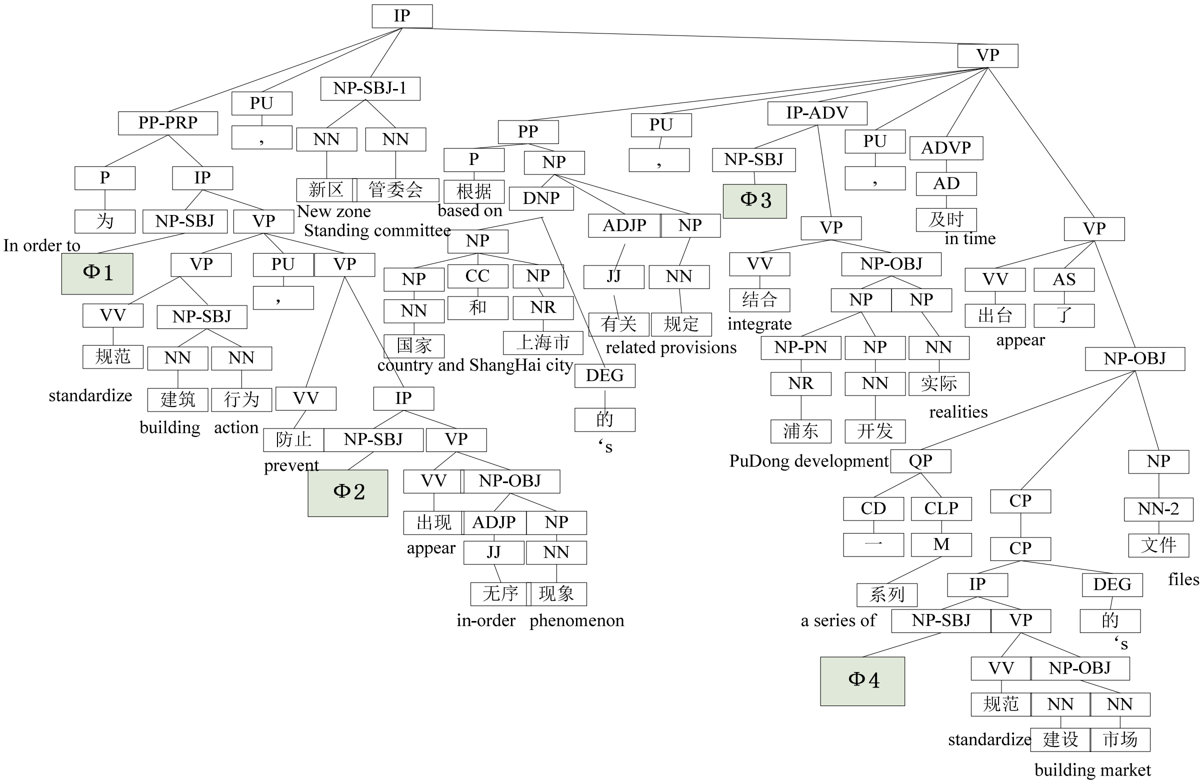


Figure 1: An example sentence from CTB 6.0, which contains four zero anaphors

(the example is : 为规范建筑行为，防止出现无序现象，新区管委会根据国家和上海市的有关规定，结合浦东开发实际，及时出台了一系列规范建设市场的文件/ In order to standardize the building action and prevent the inorder phenomenon, the standing committee of new zone annouced a series of files to standardize building market based on the related provisions of China and Shanghai in time, and the realities of the development of Pudong are considered.)

3.2 Zero anaphora corpus in Chinese

Due to lack of an available zero anaphora corpus for research purpose, we develop a Chinese zero anaphora corpus of 100 documents from CTB 6.0, which adds a layer of annotation to the manually-parsed sentences. Hoping the public availability of this corpus can push the research of zero anaphora resolution in Chinese and other languages.

```
(IP  (PP-PRP (P 为)
      (IP (NP-SBJ (-NONE- *PRO*))
         (VP (VP (VV 规范)
               (NP-OBJ (NN 建筑)
                        (NN 行为)))
     (PU ，)
     (VP (VV 防止)
         (IP (NP-SBJ (-NONE- *pro*))
            (VP (VV 出现)
                (NP-OBJ (ADJP (JJ 无序))
                         (NP (NN 现象)))))))))
```

Figure 2: An example sentence annotated in CTB 6.0

ID	Category	Description	AZAs	ZAs
1	-NONE- *T*	Used in topicalization and object preposing constructions	6	742
2	-NONE- *	Used in raising and passive constructions	1	2
3	-NONE- *PRO*	Used in control structures. The *PRO* cannot be substituted by an overt constituent.	219	399
4	-NONE- *pro*	for dropped subject or object.	394	449
5	-NONE- *RNR*	Used for right node raising (Cataphora)	0	36
6	Others	Other unknown empty categories	92	92
Total (100 documents, 35089 words)			712	1720

Table 1: Statistics on different categories of zero anaphora (AZA and ZA indicates anaphoric zero anaphor and zero anaphor respectively)

Figure 2 illustrates an example sentence annotated in CTB 6.0, where the special tag “-NONE-” represents a null constituent and thus the occurrence of a zero anaphor. In our developed corpus, we need to annotate anaphoric zero anaphors using those null constituents with the special tag of “-NONE-”.

Table 1 gives the statistics on all the six categories of zero anaphora. Since we do not consider zero cataphora in the current version, we simply redeem them non-anaphoric. It shows that among 1720 zero anaphors, only 712 (about 40%) are anaphoric. This suggests the importance of anaphoricity determination in zero anaphora resolution. Table 3 further shows that, among 712 anaphoric zero anaphors, 598 (84%) are intra-sentential and no anaphoric zero anaphors have their antecedents occurring two sentences before.

Sentence distance	AZAs
0	598
1	114
>=2	0

Table 3 Distribution of anaphoric zero anaphors over sentence distances

Figure 3 shows an example in our corpus corresponding to Figure 2. For a non-anaphoric zero anaphor, we replace the null constituent with “E-***i*** ***NZA***”, where ***i*** indicates the category of zero anaphora, with “1” referring to “-NONE *T*” etc. For an anaphoric zero anaphor, we replace it with “E-***x-y-z-i*** ***AZA***”, where ***x*** indicates the sentence id of its antecedent, ***y*** indicates the position of the first word of its antecedent in the sentence, ***z*** indicates the position of the last word of its antecedent in the sentence, and ***i*** indicates the category id of the null constituent.

```
(IP  (PP-PRP (P 为)
     (IP (NP-SBJ (E-3  NZP))
        (VP (VP (VV 规范)
              (NP-OBJ (NN 建筑)
                         (NN 行为)))
          (PU ，)
          (VP (VV 防止)
             (IP (NP-SBJ (E-7-1-1-4  AZP))
                (VP (VV 出现)
                     (NP-OBJ (ADJP (JJ 无序))
                              (NP (NN 现象))))))))
```

Figure 3: an example sentence annotated in our corpus

4 Tree Kernel-based Framework

This section presents the tree kernel-based unified framework for all the three sub-tasks in zero anaphora resolution. For each sub-task, different parse tree structures are constructed. In particular, the context-sensitive convolution tree kernel, as proposed in Zhou et al. (2008), is employed to compute the similarity between two parse trees via the SVM toolkit SVMLight.

In the tree kernel-based framework, we perform the three sub-tasks, zero anaphor detection, anaphoricity determination and antecedent identification in a pipeline manner. That is, given a zero anaphor candidate Z, the zero anaphor detector is first called to determine whether Z is a zero anaphor or not. If yes, the anaphoricity determiner is then invoked to determine whether Z is an anaphoric zero anaphor. If yes, the antecedent identifier is finally awaked to determine its antecedent. In the future work, we will explore better ways of integrating the three sub-tasks (e.g. joint learning).

4.1 Zero anaphor detection

At the first glance, it seems that a zero anaphor can occur between any two constituents in a parse tree. Fortunately, an exploration of our corpus shows that a zero anaphor always occurs just before a predicate[1] phrase node (e.g. VP). This phenomenon has also been employed in Zhao and Ng (2007) in generating zero anaphor candidates. In particular, if the predicate phrase node occurs in a coordinate structure or is modified by an adverbial node, we only need to consider its parent. As shown in Figure 1, zero anaphors may occur immediately to the left of规范/guide, 防止/avoid, 出现/appear, 根据/according to, 结合/combine, 出台/promulgate, which cover the four true zero anaphors. Therefore, it is simple but reliable in applying above heuristic rules to generate zero anaphor candidates.

Given a zero anaphor candidate, it is critical to construct a proper parse tree structure for tree kernel-based zero anaphor detection. The intuition behind our parser tree structure for zero anaphor detection is to keep the competitive information

[1] The predicate in Chinese can be categorized into verb predicate, noun predicate and preposition predicate. In our corpus, about 93% of the zero anaphors are driven by verb predicates. In this paper, we only explore zero anaphors driven by verb predicates.

about the predicate phrase node and the zero anaphor candidate as much as possible. In particular, the parse tree structure is constructed by first keeping the path from the root node to the predicate phrase node and then attaching all the immediate verbal phrase nodes and nominal phrase nodes. Besides, for the sub-tree rooted by the predicate phrase node, we only keep those paths ended with verbal leaf nodes and the immediate verbal and nominal nodes attached to these paths. Figure 4 shows an example of the parse tree structure corresponding to Figure 1 with the zero anaphor candidate Φ2 in consideration.

During training, if a zero anaphor candidate has a counterpart in the same position in the golden standard corpus (either anaphoric or non-anaphoric), a positive instance is generated. Otherwise, a negative instance is generated. During testing, each zero anaphor candidate is presented to the learned zero anaphor detector to determine whether it is a zero anaphor or not. Besides, since a zero anaphor candidate is generated when a predicate phrase node appears, there may be two or more zero anaphor candidates in the same position. However, there is normally one zero anaphor in the same position. Therefore, we just select the one with maximal confidence as the zero anaphor in the position and ignore others, if multiple zero anaphor candidates occur in the same position.

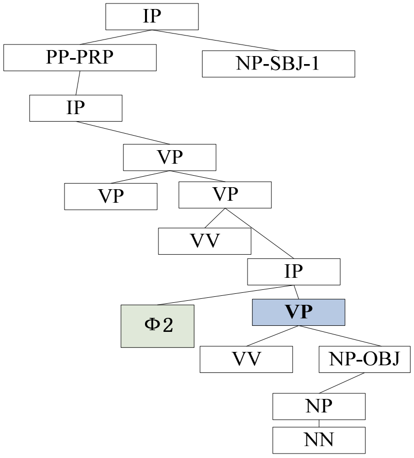


Figure 4: An example parse tree structure for zero anaphor detection with the predicate phrase node and the zero anaphor candidate Φ2 in black

4.2 Anaphoricity determination

To determine whether a zero anaphor is anaphoric or not, we limit the parse tree structure between the previous predicate phrase node and the following predicate phrase node. Besides, we only keep those verbal phrase nodes and nominal phrase nodes. Figure 5 illustrates an example of the parse tree structure for anaphoricity determination, corresponding to Figure 1 with the zero anaphor Φ2 in consideration.

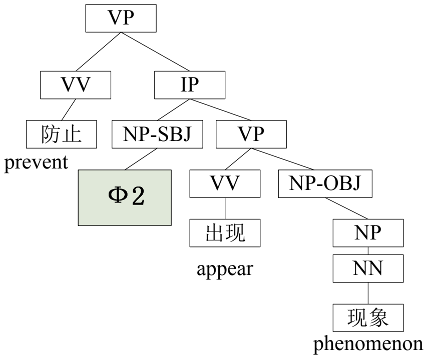


Figure 5: An example parse tree structure for anaphoricity determination with the zero anaphor Φ2 in consideration

4.3 Antecedent identification

To identify an antecedent for an anaphoric zero anaphor, we adopt the Dynamic Expansion Tree, as proposed in Zhou et al. (2008), which takes predicate- and antecedent competitor-related information into consideration. Figure 6 illustrates an example parse tree structure for antecedent identification, corresponding to Figure 1 with the anaphoric zero anaphor Φ 2 and the antecedent candidate “建筑行为/building action” in consideration.

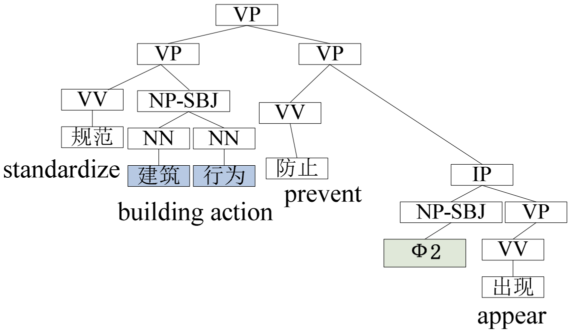


Figure 6: An example parse tree structure for antecedent identification with the anaphoric zero anaphor Φ2 and the antecedent candidate “建筑行为/building action” in consideration

In this paper, we adopt a similar procedure as Soon et al. (2001) in antecedent identification. Besides, since all the anaphoric zero anaphors have their antecedents at most one sentence away, we only consider antecedent candidates which are at most one sentence away. In particular, a document-level parse tree for an entire document is constructed by attaching the parse trees of all its sentences to a new-added upper node, as done in Yang et al. (2006), to deal with inter-sentential ones.

5 Experimentation and Discussion

We have systematically evaluated our tree kernel-based unified framework on our developed Chinese zero anaphora corpus, as described in Section 3.2. Besides, in order to focus on zero anaphor resolution itself and compare with related work, all the experiments are done on golden parse trees provided by CTB 6.0. Finally, all the performances are achieved using 5-fold cross validation.

5.1 Experimental results

Zero anaphor detection

Table 4 gives the performance of zero anaphor detection, which achieves 70.05%, 83.24% and 76.08 in precision, recall and F-measure, respectively. Here, the lower precision is much due to the simple heuristic rules used to generate zero anaphors candidates. In fact, the ratio of positive and negative instances reaches about 1:12. However, this ratio is much better than that (1:30) using the heuristic rule as described in Zhao and Ng (2007). It is also worth to point out that lower precision higher recall is much beneficial than higher precision lower recall as higher recall means less filtering of true zero anaphors and we can still rely on anaphoricity determination to filter out those false zero anaphors introduced by lower precision in zero anaphor detection.

P%	R%	F
70.05	83.24	76.08

Table 4: Performance of zero anaphor detection

Anaphoricity determination

Table 5 gives the performance of anaphoricity determination. It shows that anaphoricity determination on golden zero anaphors achieves very good performance of 89.83%, 84.21% and 86.93 in precision, recall and F-measure, respectively, although useful information, such as gender and number, is not available in anaphoricity determination. This indicates the critical role of the structural information in anaphoricity determination of zero anaphors. It also shows that anaphoricity determination on automatic zero anaphor detection achieves 77.96%, 53.97% and 63.78 in precision, recall and F-measure, respectively. In comparison with anaphoricity determination on golden zero anaphors, anaphoricity determination on automatic zero anaphor detection lowers the performance by about 23 in F-measure. This indicates the importance and the necessity for further research in zero anaphor detection.

	P%	R%	F
golden zero anaphors	89.83	84.21	86.93
zero anaphor detection	77.96	53.97	63.78

Table 5: Performance of anaphoricity determination

Antecedent identification

Table 6 gives the performance of antecedent identification given golden zero anaphors. It shows that antecedent identification on golden anaphoric zero anaphors achieves 88.93%, 68.36% and 77.29 in precision, recall and F-measure, respectively. It also shows that antecedent identification on automatic anaphoricity determination achieves 80.38%, 47.28% and 59.24 in precision, recall and F-measure, respectively, with a decrease of about 8% in precision, about 21% in recall and about 18% in F-measure, in comparison with antecedent identification on golden anaphoric zero anaphors. This indicates the critical role of anaphoricity determination in antecedent identification.

	P%	R%	F
golden anaphoric zero anaphors	88.90	68.36	77.29
anaphoricity determination	80.38	47.28	59.54

Table 6: Performance of antecedent identification given golden zero anaphors

Overall: zero anaphora resolution

Table 7 gives the performance of overall zero anaphora resolution with automatic zero anaphor detection, anaphoricity determination and antecedent identification. It shows that our tree kernel-based framework achieves 77.66%, 31.74% and 45.06 in precision, recall and F-measure. In comparison with Table 6, it shows that the errors caused by automatic zero anaphor detection decrease the performance of overall zero anaphora resolution by about 14 in F-measure, in comparison with golden zero anaphors.

P%	R%	F
77.66	31.74	45.06

Table 7: Performance of zero anaphora resolution

Figure 7 shows the learning curve of zero anaphora resolution with the increase of the number of the documents in experimentation, with the horizontal axis the number of the documents used and the vertical axis the F-measure. It shows that the F-measure is about 42.5 when 20 documents are used in experimentation. This figure increases very fast to about 45 when 50 documents are used while further increase of documents only slightly improves the performance.

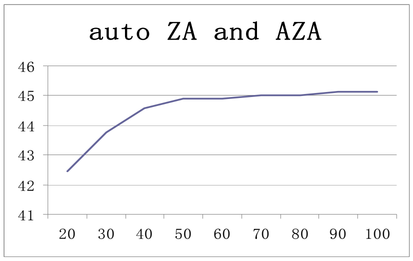


Figure 7: Learning curve of zero anaphora resolution over the number of the documents in experimentation

Table 8 shows the detailed performance of zero anaphora resolution over different sentence distance between a zero anaphor and its antecedent. It is expected that both the precision and the recall of intra-sentential resolution are much higher than those of inter-sentential resolution, largely due to the much more dependency of intra-sentential antecedent identification on the parse tree structures.

Sentence distance	P%	R%	F
0	85.12	33.28	47.85
1	46.55	23.64	31.36
2	-	-	-

Table 8: Performance of zero anaphora resolution over sentence distances

Table 9 shows the detailed performance of zero anaphora resolution over the two major zero anaphora categories, "-NONE- *PRO*" and "-NONE- *pro*". It shows that our tree kernel-based framework achieves comparable performance on them, both with high precision and low recall. This is in agreement with the overall performance.

ID	Category	P%	R%	F
3	-NONE- *PRO*	79.37	34.23	47.83
4	-NONE- *pro*	77.03	30.82	44.03

Table 9: Performance of zero anaphora resolution over major zero anaphora categories

5.2 Comparison with previous work

As a representative in Chinese zero anaphora resolution, Zhao and Ng (2007) focused on anaphoricity determination and antecedent identification using feature-based methods. In this subsection, we will compare our tree kernel-based framework with theirs in details.

Corpus

Zhao and Ng (2007) used a private corpus from Converse (2006). Although their corpus contains 205 documents from CBT 3.0, it only deals with the zero anaphors under the zero anaphora category of "-NONE- *pro*" for dropped subjects/objects. Furthermore, Zhao and Ng (2007) only considered zero anaphors with explicit noun phrase referents and discarded zero anaphors with split antecedents (i.e. split into two separate noun phrases) or referring to entities. As a result, their corpus is only about half of our corpus in the number of zero anaphors and anaphoric zero anaphors. Besides, our corpus deals with all the types of zero anaphors and all the categories of zero anaphora except zero cataphora.

Method

Zhao and Ng (2007) applied feature-based methods on anaphoricity determination and antecedent identification with most of features structural in nature. For zero anaphor detection, they used a very simple heuristic rule to generate zero anaphor candidates. Although this rule can recover almost all the zero anaphors, it suffers from very low precision by introducing too many false zero anaphors and thus may lead to low performance in anaphoricity determination, much due to the imbalance between positive and negative training examples with the ratio up to about 1:30.

In comparison, we propose a tree kernel-based unified framework for all the three sub-tasks in zero anaphora resolution. In particular, different parse tree structures are constructed for different sub-tasks. Besides, a context sensitive convolution tree kernel is employed to directly compute the similarity between the parse trees.

For fair comparison with Zhao and Ng (2007), we duplicate their system and evaluate it on our developed Chinese zero anaphora corpus, using the same J48 decision tree learning algorithm in Weka and the same feature sets for anaphoricity determination and antecedent identification.

Table 10 gives the performance of the feature-based method, as described in Zhao and Ng (2007), in anaphoricity determination on our developed corpus. In comparison with the tree kernel-based method in this paper, the feature-based method performs about 16 lower in F-measure, largely due to the difference in precision (63.61% vs 89.83%), when golden zero anaphors are given. It also shows that, when our tree kernel-based zero anaphor detector is employed [2], the feature-based method gets much lower precision with a gap of about 31%, although it achieves slightly higher recall.

	P%	R%	F
golden zero anaphors	63.61	79.71	70.76
zero anaphor detection	46.17	57.69	51.29

Table 10: Performance of the feature-based method (Zhao and Ng 2007) in anaphoricity determination on our developed corpus

	P%	R%	F
golden anaphoric zero anaphors	77.45	51.97	62.20
golden zero anaphpors and feature-based anaphoricity determination	75.17	29.69	42.57
overall: tree kernel-based zero anaphor detection and feature-based anaphoricity determination	70.67	23.64	35.43

Table 11: Performance of the feature-based method (Zhao and Ng 2007) in antecedent identification on our developed corpus

Table 11 gives the performance of the feature-based method, as described in Zhao and Ng (2007), in antecedent identification on our developed corpus. In comparison with our tree kernel-based method, it shows that 1) when using golden anaphoric zero anaphors, the feature-based method performs about 11%, 17% and 15 lower in precision, recall and F-measure, respectively; 2) when golden zero anaphors are given and feature-based anaphoricity determination is applied, the feature-based method performs about 5%, 18% and 17 lower in precision, recall and F-measure, respectively; and 3) when tree kernel-based zero anaphor detection and feature-based anaphoricity determination are applied, the feature-based method performs about 7%, 8% and 10 lower in precision, recall and F-measure, respectively.

In summary, above comparison indicates the critical role of the structural information in zero anaphora resolution, given the fact that most of features in the feature-based methods in Zhao and Ng (2007) are also structural, and the necessity of tree kernel methods in modeling such structural information, even if more feature engineering in the feature-based methods may improve the performance to a certain extent.

6 Conclusion and Further Work

This paper proposes a tree kernel-based unified framework for zero anaphora resolution, which can be divided into three sub-tasks: zero anaphor detection, anaphoricity determination and antecedent identification.

The major contributions of this paper include: 1) We release a wide-coverage Chinese zero anaphora corpus of 100 documents, which adds a layer of annotation to the manually-parsed sentences in the Chinese Treebank (CTB) 6.0. 2) To our best knowledge, this is the first systematic work dealing with all the three sub-tasks in Chinese zero anaphora resolution via a unified framework. 3) Employment of tree kernel-based methods indicates the critical role of the structural information in zero anaphora resolution and the necessity of tree kernel methods in modeling such structural information.

In the future work, we will systematically evaluate our framework on automatically-generated parse trees, construct more effective parse tree structures for different sub-tasks of zero anaphora resolution, and explore joint learning among the three sub-tasks.

Besides, we only consider zero anaphors driven by a verb predicate phrase node in this paper. In the future work, we will consider other situations. Actually, among the remaining 7% zero anaphors, about 5% are driven by a preposition phrase (PP) node, and 2% are driven by a noun phrase (NP) node. However, our preliminary experiments show that simple inclusion of those PP-driven and NP-driven zero anaphors will largely increase the imbalance between positive and negative instances, which significantly decrease the performance.

Finally, we will devote more on further developing our corpus, with the ultimate mission of annotating all the documents in CBT 6.0.

[2] We do not apply the simple heuristic rule, as adopted in Zhao and Ng (2007), in zero anaphor detection, due to its much lower performance, for fair comparison on the other two subtsaks..

Acknowledgments

This research was supported by Projects 60873150, 90920004 and 61003153 under the National Natural Science Foundation of China.

References

S. Converse. 2006. *Pronominal Anaphora Resolution in Chinese*. Ph.D. Thesis, Department of Computer and Information Science. University of Pennsylvania.

M. Collins and N. Duffy. 2001. *Convolution kernels for natural language*. NIPS’2001:625-632.

A. Ferrandez and J. Peral. 2000. *A computational approach to zero-pronouns in Spanish*. ACL'2000:166-172.

R. Iida, K. Inui, and Y. Matsumoto. 2006. *Exploiting syntactic patterns as clues in zero-anaphora resolution*. COLING-ACL'2006:625-632

H. Isozaki and T. Hirao. 2003. *Japanese zero pronoun resolution based on ranking rules and machine learning*. EMNLP'2003:184-191

F. Kong, G.D. Zhou and Q.M. Zhu. 2009 Employing the Centering Theory in Pronoun Resolution from the Semantic Perspective. EMNLP’2009: 987-996

C. N. Li and S. A. Thompson. 1979. *Third-person pronouns and zero-anaphora in Chinese discourse*. Syntax and Semantics, 12:311-335.

W. Li. 2004. *Topic chains in Chinese discourse*. Discourse Processes, 37(1):25-45.

A. Moschitti. 2004. *A Study on Convolution Kernels for Shallow Semantic Parsing*, ACL’2004.

L.H. Qian, G.D. Zhou, F. Kong, Q.M. Zhu and P.D. Qian. 2008. *Exploiting constituent dependencies for tree kernel-based semantic relation extraction*. COLING’2008:697-704

K. Seki, A. Fujii, and T. Ishikawa. 2002. *A probabilistic method for analyzing Japanese anaphora intergrating zero pronoun detection and resolution*. COLING'2002:911-917

R. Sasano. D. Kawahara and S. Kurohashi. 2008. *A fully-lexicalized probabilistic model for Japanese zero anaphora resolution*. COLING'2008:769-776

W.M. Soon, H.T. Ng and D. Lim. 2001. *A machine learning approach to coreference resolution of noun phrase*. Computational Linguistics, 2001, 27(4):521-544.

V. Ng and C. Cardie 2002. Improving machine learning approaches to coreference resolution. ACL’2002: 104-111

X.F. Yang, G.D. Zhou, J. Su and C.L. Chew. 2003. Coreference Resolution Using Competition Learning Approach. ACL’2003:177-184

X.F. Yang, J. Su and C.L. Tan 2008. A Twin-Candidate Model for Learning-Based Anaphora Resolution. Computational Linguistics 34(3):327-356

N. Xue, F. Xia, F.D. Chiou and M. Palmer. 2005. *The Penn Chinese TreeBank: Phrase structure annotation of a large corpus.* Natural Language Engineering, 11(2):207-238.

X.F. Yang, J. Su and C.L. Tan. 2006. *Kernel-based pronoun resolution with structured syntactic knowledge*. COLING-ACL'2006:41-48.

D. Zelenko, A. Chinatsu and R. Anthony. 2003. *Kernel methods for relation extraction.* Journal of Machine Learning Research, 3(2003):1083-1106

S. Zhao and H.T. Ng. 2007. *Identification and Resolution of Chinese Zero Pronouns: A Machine Learning Approach.* EMNLP-CoNLL'2007:541-550.

S. Zhao and R. Grishman. 2005. *Extracting relations with integrated information using kernel methods.* ACL’2005:419-426

G.D. Zhou, F. Kong and Q.M. Zhu. 2008. *Context-sensitive convolution tree kernel for pronoun resolution.* IJCNLP'2008:25-31

G.D. Zhou, M. Zhang, D.H. Ji and Q.M. Zhu. 2007. *Tree kernel-based relation extraction with context-sensitive structured parse tree information*. EMNLP-CoNLL’2007:728-736

Automatic Comma Insertion for Japanese Text Generation

Masaki Murata
Graduate School of
Information Science,
Nagoya University, Japan
murata@el.itc.nagoya-u.ac.jp

Tomohiro Ohno
Graduate School of
International Development,
Nagoya University, Japan
ohno@nagoya-u.jp

Shigeki Matsubara
Graduate School of
Information Science,
Nagoya University, Japan
matubara@nagoya-u.jp

Abstract

This paper proposes a method for automatically inserting commas into Japanese texts. In Japanese sentences, commas play an important role in explicitly separating the constituents, such as words and phrases, of a sentence. The method can be used as an elemental technology for natural language generation such as speech recognition and machine translation, or in writing-support tools for non-native speakers. We categorized the usages of commas and investigated the appearance tendency of each category. In this method, the positions where commas should be inserted are decided based on a machine learning approach. We conducted a comma insertion experiment using a text corpus and confirmed the effectiveness of our method.

1 Introduction

In Japanese sentences, commas are inserted to mark word boundaries that might be otherwise unclear because Japanese is a non-segmented language. They are also inserted at sharp semantic boundaries to improve the readability of a sentence. While there is a tendency about the positions where commas should be inserted in a Japanese sentence, there is no clear standard for these positions. Therefore, it is hard for non-natives of Japanese such as foreign students to insert commas properly, and the method for automatic comma insertion is required to support sentence generation by such people. In addition, this method is expected to be useful for improving readability of texts generated by automatic speech recognition or machine translation.

This paper proposes a method for automatically inserting commas into Japanese texts. There are several usages of commas, and the positions to insert commas depend on these usages. Therefore, we grouped the usages of commas into nine categories, and investigated the appearance tendency for each category to find the effective features of machine learning by using Japanese newspaper articles. Based on the analysis of comma positions, our method decides whether or not to insert a comma at each *bunsetsu*[1] boundary in an input sentence by machine learning.

We conducted an experiment on comma insertion using the Kyoto Text Corpus (Kurohashi and Nagao, 1998), and obtained higher recall and precision than those of the baseline, leading us to confirm the effectiveness of our method.

This paper is organized as follows: The next section presents related works. Section 3 gives preliminary analyses. Section 4 explains how our comma insertion method works. An experiment and discussions are presented in Sections 5 and 6, respectively.

2 Related Works

There have been many investigations on comma insertion into output texts of speech recognition systems to improve the readability (Christensen et al., 2001; Kim and Woodland, 2001; Liu et al., 2006; Shimizu et al., 2008). Their methods insert commas using pause information of speakers, based on the idea that a point at which a speaker takes a breath partly corresponds to a point where a comma is inserted. However, since pause information cannot be obtained from texts, we cannot use this approach because our targets are written texts.

In addition, there have been some investigations

[1] *Bunsetsu* is a linguistic unit in Japanese that roughly corresponds to a basic phrase in English. A bunsetsu consists of one independent word and zero or more ancillary words.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 892–901,
MIT, Massachusetts, USA, 9-11 October 2010.

on comma insertion into non-Japanese written texts (White and Rajkumar, 2008; Guo et al., 2010). In Japanese, there are several usages of commas, and some usages are specific to Japanese due to its linguistic nature. Therefore, just adopting the above mentioned methods, which have been developed to process non-Japanese texts, is not sufficient to enable high-quality comma insertion into Japanese sentences. Development of a method based on the detailed analysis of Japanese commas is required.

Furthermore, there have been some investigations on comma insertion into Japanese written texts (Hayashi, 1992; Suzuki et al., 1995). These investigations have adopted rule-based methods. However, the number of their rules is not necessarily sufficient, and no quantitative evaluation has been performed.

3 Analyses on Comma Usages

There have been several discussions on commas, including the draft of "Kutou-hou (punctuation)" made by Archives Division, Minister's Secretariat, Japanese Ministry of Education, Science and Culture in 1906. There are several usages of commas, and depending on the usage, the types of positions where commas are inserted are different. First, we examined some previous publications on commas (Honda, 1982; Inukai, 2002; Shogakukan's editiorial department, 2007). Based on the results of the examination, we classified the usages of commas into nine categories shown in Table 1. Here, commas in Japanese sentences and commas in English sentences have some common roles. In Japanese sentences, some commas have the same roles as commas in English sentences, but some commas have roles specific to Japanese due to its linguistic nature such as "Japanese is a non-segmented language" or "Japanese has *kanji* characters and *katakana* characters."

Table 1: Categorization of usages of commas

#	usage of comma
1	commas between clauses
2	commas indicating clear dependency relations
3	commas for avoiding reading mistakes and reading difficulty
4	commas indicating the subject
5	commas inserted after a conjunction or adverb at the beginning of a sentence
6	commas inserted between parallel words or phrases
7	commas inserted after an adverbial phrase to indicate time
8	commas emphasizing the adjacent word
9	other

Table 2: Size of the analysis data

sentences	11,821
bunsetsus	117,501
characters	503,970
commas	16,595
characters per sentence	42.63

In our study, positions where a comma should be inserted are detected by using machine learning. We investigated the Kyoto Text Corpus version 4.0 (Kurohashi and Nagao, 1998) to find the effective features. The Kyoto Text Corpus is a collection of Japanese articles of Mainichi newspaper. We used the articles on January 1st and from January 3rd to 11th in 1995 as the analysis data. Table 2 shows the size of the data. The data had been manually annotated with information on morphological analysis, bunsetsu segmentation and dependency[2] analysis. Clause boundaries were detected by the clause boundary detection program CBAP (Kashioka and Maruyama, 2004).

Out of all the inserted commas, only 1.43% were inserted at positions which were not bunsetsu boundaries. Therefore, we analyzed only commas inserted at bunsetsu boundaries. Of 105,680 bunsetsu boundaries, commas were inserted at 16,357 bunsetsu boundaries, that is, the rate of comma insertion was 15.48%. In the following sections, we focus on morphemes, clause boundaries, dependency relation and the number of characters between commas, and investigate their relations with commas.

3.1 Commas between Clauses

If a sentence consists of several clauses, inserting a comma between clauses makes clear the sentence

[2] A dependency in a Japanese sentence is a modification relation in which a modifier bunsetsu depends on a modified bunsetsu. That is, the modifier bunsetsu and the modified bunsetsu work as a modifier and a modifyee, respectively.

Table 3: Rates of comma insertion according to the clause boundary type

type of clause boundary	ratio of comma insertion (%)	
topicalized element-*wa*	16.94	(1,446/8,536)
adnominal clause	0.72	(43/5,960)
continuous clause	84.57	(2,685/3,175)
compound clause-*te*	23.31	(394/1,690)
quotational clause	4.40	(74/1,680)
supplement clause	17.53	(245/1,398)
discourse marker	60.13	(650/1,081)
compound clause-*ga*	93.85	(946/1,008)
compound clause-*de*	84.52	(606/717)
condition clause-*to*	81.66	(423/518)

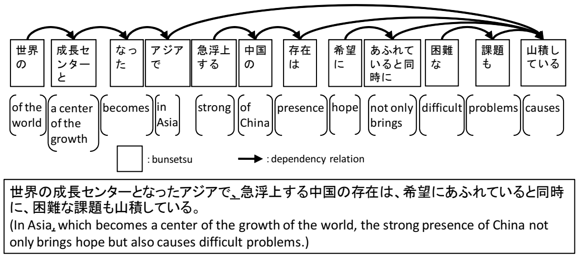
structure. Therefore, a clause boundary is considered to be a strong candidate of a position where a comma is inserted. For example, in the following sentence[3]:

- 国連による対イラク制裁解除に向け、関係の深い仏に一層の協力を求めるのが狙いとみられる。(Toward lifting the sanctions imposed on Iraq by United Nations, the aim seems to be to request further cooperation from France, which has close ties to Iraq.)

a comma is inserted at the clause boundary right after the continuous clause "国連による対イラク制裁解除に向け (Toward lifting the sanctions imposed on Iraq by United Nations)." Like this example, the same usage of commas is seen in English as well.

In the analysis data, there existed 29,278 clause boundaries excluding sentence breaks. Among them, commas were inserted at 8,805 positions (30.01%). The rate is higher than that of bunsetsu boundaries. This indicates that commas tend to be inserted at clause boundaries.

We investigated the rate of comma insertion about 114 types[4] of clause boundaries. Table 3 shows the top 10 clause boundary types according to the occurrence frequency, and the rates of comma insertion. In cases of "continuous clause" and "compound clause-*de*," the rates were higher than 84%. On the other hand, in cases of "adnominal clause" and "quotational clause," the rates were lower than 5%. This means that the likelihoods of comma insertion are different according to the clause boundary type.



世界の成長センターとなったアジアで、急浮上する中国の存在は、希望にあふれていると同時に、困難な課題も山積している。
(In Asia, which becomes a center of the growth of the world, the strong presence of China not only brings hope but also causes difficult problems.)

Figure 1: Commas making clear dependency relations

3.2 Commas and Dependency Structure

Commas have a role to make dependency relations clearer. Commas tend to be inserted right after a bunsetsu that depends on a distant bunsetsu. In Figure 1, although the bunsetsu "アジアで (in Asia)" depends on the bunsetsu "山積している (causes)," if the comma right after the bunsetsu "アジアで (in Asia)" is not inserted, the readers might mistakenly understand that the bunsetsu "アジアで (in Asia)" depends on the next bunsetsu "急浮上する (strong)." To avoid the mistake, the comma is inserted.

In the analysis data, there existed 66,984 bunsetsus which depend on the next bunsetsu. Among the bunsetsu boundaries right after them, 2,302 (3.44%) were the positions where a comma was inserted. On the other hand, in the case of a bunsetsu boundary right after a bunsetsu which does not depend on the next bunsetsu, the rate of comma insertion was 36.32% (14,055/38,696).

In addition, when the modifyee of a bunsetsu is located outside the clause containing the bunsetsu, i.e. to the right of the clause end, commas are considered to be more frequently inserted right after the bunsetsu because such bunsetsu causes more complex dependency structure. The rate of comma insertion right after such bunsetsu is 54.24%.

[3] We underlined commas which we mentioned in the example and the corresponding positions in the translation of the example.

[4] In our research, we used the types of clause boundaries defined by the Clause Boundary Annotation Program (Kashioka and Maruyama, 2004).

3.3 Commas for Avoiding Reading Mistakes and Reading Difficulty

Although, unlike English, Japanese is a non-segmented language, word boundaries are easy to detect because Japanese has three types of characters; *hiragana* characters, *katakana* characters, and *kanji* characters. However, if the same types of characters appear sequentially, readers may make a reading mistake or feel difficulty in reading them. To avoid such mistakes and difficulty, there is a usage of commas specific to Japanese.

In the following example, a comma is inserted between two sequentially appearing words "焼却 (burned)" and "灰 (ashes)" both of which consist of only *kanji* characters.

- 川崎さんの遺体を群馬県利根郡片品村花咲の知人宅に運んで焼却、灰を同村の山林に捨てたことを認める内容という。(He seemed to acknowledge that he had carried the corpse of Mr. Kawasaki to an acquaintance in Hanasaki, Katashina-mura, Tone-gun, Gunma Prefecture, burned it_and abandoned its ashes in the mountain forest in Katashina-mura.)

The comma was inserted because if there was no comma, the word boundary would become unclear and reading difficulty would be caused. Among 2,409 bunsetsu boundaries over which *kanji* characters appeared sequentially, commas were inserted at 2,188 (90.83%) bunsetsu boundaries. In the case of *katakana* characters, the rate was 97.69% (211/216). Commas tend to be inserted at most bunsetsu boundaries if *kanji* characters or *katakana* characters sequentially appear over a boundary.

3.4 Commas Indicating the Subject

Commas are considered to be inserted right after a bunsetsu that represents the subject of a sentence. For example, in Figure 2, a comma is inserted right after the bunsetsu "戦火は (war)" to indicate that the bunsetsu is the subject of the sentence. Here, we pay attention to the clause boundary of the type "topicalized element-*wa*." The rate that commas were inserted at the clause boundaries "topicalized element-*wa*" was 16.94% (1,446/8,536). This rate is almost the same as that of bunsetsu boundaries. On the other hand, the commas inserted at the clause boundaries "topicalized element-*wa*" accounted for 8.84% (1,446/16,357) of all the inserted commas.

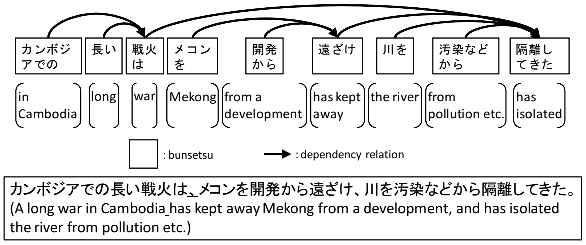


Figure 2: Comma insertion at the clause boundary "topicalized element-*wa*"

In the case of the clause boundary "topicalized element-*wa*" right after a bunsetsu which does not depend on the next bunsetsu (e.g., the bunsetsu "戦火は (war)" in Figure 2), the rate of comma insertion was 20.71% (1,426/6,886). The rate is higher than that of all the clause boundaries "topicalized element-*wa*." This shows that commas tend to be especially inserted at the "topicalized element-*wa*" right after bunsetsus which do not depend on the next bunsetsu.

3.5 Commas after Conjunction or Adverb

Commas tend to be inserted right after a conjunction or an adverb located at the beginning of a sentence. These commas correspond to English commas which are inserted right after a word such as "however" and "furthermore" located at the beginning of a sentence.

- しかし、私はそれに同感する気持ちになれない。(However, I do not feel like agreeing on it.)

In the analysis data, there existed 695 bunsetsus whose rightmost morpheme is a conjunction and which are located at the beginning of a sentence. Among them, commas were inserted right after 498 (71.65%) bunsetsus. In the case of bunsetsus whose rightmost morpheme is an adverb, the rate was 30.97% (140/452).

3.6 Commas Inserted between Parallel Words or Phrases

Commas have a function which makes clear separation between parallel words or phrases. The following example shows commas separating parallel nouns.

- むしろ地球規模の環境、人口、食糧など広範に国連の果たさなければならない役割は大きい。(The United Nations should play a lot of roles in a broad range of fields, such as the global environment, population, and food.)

In this example, commas are inserted to separate parallel nouns "環境 (environment)," "人口 (population)" and "食糧 (food)". In English, there are commas which perform the same role. In fact, commas were inserted between "environment" and "population" and between "population" and "food" in the translation of the above example. When bunsetsus whose rightmost morpheme is a noun appear sequentially, the rate of comma insertion between such bunsetsus is 59.39% (3,330/5,607).

Also, commas are inserted to separate parallel phrases. In the following example,

- メニューは前夜、首相が何を食べたかを調べて同じ献立を避けたり、和食と洋食のバランスを考えたりして決める。(The menu is decided by avoiding the menu the Prime Minister ate on the previous night, and by considering the balance between the Japanese food and the European food.)

a comma is inserted right after the bunsetsu "避けたり (avoiding)" to make clear separation between the parallel phrases "同じ献立を避けたり (by avoiding the menu)" and "和食と洋食のバランスを考えたりして (by considering the balance between the Japanese food and the European food)." The rate of comma insertion between two parallel phrases is 79.89% (751/940). This is much higher than that of bunsetsu boundaries, indicating that commas tend to be inserted when phrases are paralleled.

3.7 Number of Characters between Commas

If there are too many commas at a short distance, the sentence becomes hard to read. Therefore, the number of characters between commas is expected to be not too small. Also, because a long sequence of characters without a comma is generated if the distance between commas is very long, the occurrence frequency of such sequences of characters is considered to be low.

We investigated the number of characters between commas and its occurrence frequency. Figure 3 shows the results of investigation. When the number of characters between commas is either large or small, the occurrence frequency is low.

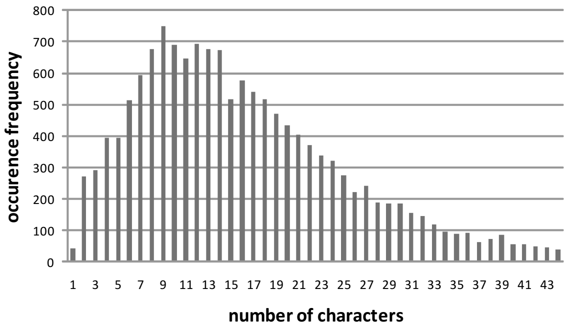


Figure 3: Number of characters between commas and its occurrence frequency

4 Comma Insertion Method

In our method, a sentence, on which morphological analysis, bunsetsu segmentation, clause boundary analysis and dependency analysis have been performed, is considered the input. Our method decides whether or not to insert a comma at each bunsetsu boundary in an input sentence. Based on the analysis results in Section 3, our method adopts the bunsetsu boundaries as candidate positions where a comma is inserted. Our method identifies the most appropriate combination among all combinations of positions where a comma can be inserted, by using the probabilistic model. In this paper, input sentences which consist of n bunsetsus are represented by $B = b_1 \cdots b_n$, and the results of comma insertion by $R = r_1 \cdots r_n$. Here, r_i is 1 if a comma is inserted right after bunsetsu b_i, and 0 otherwise. We indicate the j-th sequence of bunsetsus created by dividing an input sentence into m sequences as $L_j = b^j_1 \cdots b^j_{n_j} (1 \leq j \leq m)$, and then, $r^j_k = 0$ if $1 \leq k < n_j$, and $r^j_k = 1$ if $k = n_j$.

4.1 Probabilistic Model for Comma Insertion

When an input sentence B is provided, our method identifies the comma insertion R that maximizes the conditional probability $P(R|B)$. Assuming that whether or not to insert a comma right after a bunsetsu is independent of other commas except the

Table 4: Features used for the maximum entropy method

morphological information	the rightmost independent morpheme, i.e. head word, (part-of-speech and inflected form) and rightmost morpheme (part-of-speech) of a bunsetsu b_k^j
	the rightmost morpheme (a surface form) of b_k^j if the rightmost morpheme is a particle
	the first morpheme (part-of-speech) of b_{k+1}^j
commas inserted between clauses	whether or not a clause boundary exists right after b_k^j
	type of a clause boundary right after b_k^j if there exists a clause boundary
commas indicating clear dependency relations	whether or not b_k^j depends on the next bunsetsu
	whether or not b_k^j depends on a bunsetsu located after the final bunsetsu of the clause including the next bunsetsu of b_k^j
	whether or not b_k^j is depended on by the bunsetsu located right before it
	whether or not the dependency structure of a sequence of bunsetsus between b_k^j and b_1^j is closed
commas avoiding reading mistakes and reading difficulty	whether or not both the rightmost morpheme of b_k^j and first morpheme of b_{k+1}^j are *kanji* characters
	whether or not both the rightmost morpheme of b_k^j and first morpheme of b_{k+1}^j are *katakana* characters
commas indicating the subject	whether or not there exists a clause boundary "topicalized element-*wa*" right after b_k^j and b_k^j depends on the next bunsetsu
	whether or not there exists a clause boundary "topicalized element-*wa*" right after b_k^j and the string of characters right before b_k^j is "では (*dewa*)"
	the number of characters in a phrase indicating the subject[5] if there exists a clause boundary "topicalized element-*wa*" right after b_k^j
	whether or not a clause boundary "topicalized element-*wa*" exists right after b_k^j and a bunsetsu whose rightmost morpheme is a verb depends on the modified bunsetsu of b_k^j
commas inserted after a conjunction or adverb at the beginning of a sentence	whether or not b_k^j appears at the beginning of a sentence and its rightmost morpheme is a conjunction
	whether or not b_k^j appears at the beginning of a sentence and its rightmost morpheme is an adverb
commas inserted between parallel words or phrases	whether or not both the rightmost morphemes of b_k^j and b_{k+1}^j are nouns
	whether or not a predicate at the sentence end is depended on by b_k^j whose rightmost independent morpheme is a verb and by any of the bunsetsus which are located after b_k^j and of which the rightmost independent morpheme is a verb
number of characters from b_1^j to b_k^j	one of the following 4 categories if the number of characters from b_1^j to b_k^j is found there ([num = 1], [2 ≤ num ≤ 3], [4 ≤ num ≤ 21], [22 ≤ num])

one appearing immediately before that bunsetsu, $P(R|B)$ can be calculated as follows:

$$\begin{aligned}
& P(R|B) \\
=& P(r_1^1=0,\cdots,r_{n_1-1}^1=0,r_{n_1}^1=1,\cdots, \\
& \qquad r_1^m=0,\cdots,r_{n_m-1}^m=0,r_{n_m}^m=1|B) \\
\cong & P(r_1^1=0|B)\times\cdots \\
& \times P(r_{n_1-1}^1=0|r_{n_1-2}^1=0,\cdots,r_1^1=0,B) \\
& \times P(r_{n_1}^1=1|r_{n_1-1}^1=0,\cdots,r_1^1=0,B)\times\cdots \\
& \times P(r_1^m=0|r_{n_{m-1}}^{m-1}=1,B)\times\cdots \\
& \times P(r_{n_m-1}^m=0|r_{n_m-2}^m=0,\cdots,r_1^m=0,r_{n_{m-1}}^{m-1}=1,B) \\
& \times P(r_{n_m}^m=1|r_{n_m-1}^m=0,\cdots,r_1^m=0,r_{n_{m-1}}^{m-1}=1,B)
\end{aligned} \tag{1}$$

where $P(r_k^j = 1|r_{k-1}^j = 0, \cdots, r_1^j = 0, r_{n_{j-1}}^{j-1} = 1, B)$ is the probability that a comma is inserted right after a bunsetsu b_k^j when the sequence of bunsetsus B is provided and the position of j-th comma is identified. Similarly, $P(r_k^j = 0|r_{k-1}^j = 0, \cdots, r_1^j = 0, r_{n_{j-1}}^{j-1} = 1, B)$ is the probability that a comma is not inserted right after a bunsetsu b_k^j. These probabilities are estimated by the maximum entropy method. The result R which maximizes the conditional probability $P(R|B)$ is regarded as the most appropriate result of comma insertion, and calculated by dynamic programming.

4.2 Features on Maximum Entropy Method

To estimate $P(r^j_k = 1 | r^j_{k-1} = 0, \cdots, r^j_1 = 0, r^{j-1}_{n_{j-1}} = 1, B)$ and $P(r^j_k = 0 | r^j_{k-1} = 0, \cdots, r^j_1 = 0, r^{j-1}_{n_{j-1}} = 1, B)$ by the maximum entropy method, we used the features in Table 4 based on the analysis described in Section 3.

5 Experiment

To evaluate the effectiveness of our method, we conducted an experiment using a Japanese text corpus.

5.1 Outline of Experiment

As the experimental data, we used the newspaper articles in the Kyoto Text Corpus version 4.0 (Kurohashi and Nagao, 1998). We used the articles from January 14th to 17th as the test data. The training data is same as the analysis data. Table 5 shows the size of the test data. Here, we used the maximum entropy method tool (Le, 2008) with the default options except “-i 2000.”

In the evaluation, we obtained the recall, the precision and their harmonic mean, i.e., F-measure. The recall and precision are respectively defined as follows.

$$recall = \frac{\#\ of\ correctly\ inserted\ commas}{\#\ of\ commas\ in\ the\ correct\ data}$$

$$precision = \frac{\#\ of\ correctly\ inserted\ commas}{\#\ of\ inserted\ commas}$$

In our research, to realize automatic comma insertion with high quality, we analyzed each usage of commas and decided the features for the ME method based on the analysis. To confirm the effectiveness of our features, we established the baseline method as a comparative method whereby commas are inserted by the ME method in which only simple morphological information is used. The baseline method uses the morphological information in Table 4 and the information of the rightmost morpheme (a surface form) of a bunsetsu as features.

5.2 Experimental Results

Table 6 shows the experimental results of the baseline and our method. The recall and precision were 69.13% and 84.13% respectively, and we confirmed that our method had higher performance than the baseline method. The percentage of sentences wherein all commas were correctly inserted was 55.81%.

Table 5: Size of test data

sentences	4,659
bunsetsus	46,511
characters	198,899
commas	6,549
characters per sentence	42.69

Table 6: Experimental results

	recall	precision	F-measure
our method	69.13% (4,527/6,549)	84.13% (4,527/5,381)	75.90
baseline	51.38% (3,365/6,549)	70.90% (3,365/4,746)	59.58

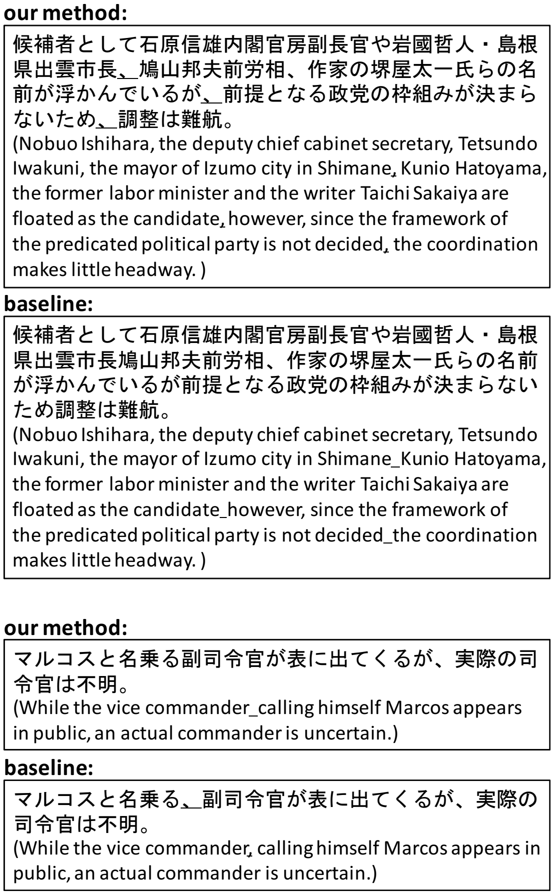
Figure 4 shows the comparison between the results of our method and the baseline method. The baseline method was not able to insert commas right after the bunsetsu “浮かんでいるが (are floated)” or “決まらないため (not decided)” but inserted commas at unnatural positions such as between “名乗る (calling himself)” and “副司令官が (the vice commander).” On the other hand, our method was able to insert commas properly at such bunsetsu boundaries.

6 Discussion

6.1 Error Analysis

Among positions where commas existed in the test data, there existed 2,022 positions where our method did not insert commas. Among them, 862 were clause boundaries, and the clause boundary “topicalized element-*wa*” accounted for 53.36% (460/862) of them. There were a lot of clause boundaries of the type “topicalized element-*wa*,” and the number of commas inserted at such boundaries was large. But, the rate of comma insertion itself was not very high. We can say that the four features about “topicalized element-*wa*” did not always work well. Ta-

[5] Phrases indicating the subject is a sequence of bunsetsus consisting of b^j_k and all bunsetsus that are connected to b^j_k when we trace their dependency relationship in modifier-to-modifyee direction.



our method:

候補者として石原信雄内閣官房副長官や岩國哲人・島根県出雲市長、_鳩山邦夫前労相、作家の堺屋太一氏らの名前が浮かんでいるが、_前提となる政党の枠組みが決まらないため、_調整は難航。
(Nobuo Ishihara, the deputy chief cabinet secretary, Tetsundo Iwakuni, the mayor of Izumo city in Shimane, Kunio Hatoyama, the former labor minister and the writer Taichi Sakaiya are floated as the candidate, however, since the framework of the predicated political party is not decided, the coordination makes little headway.)

baseline:

候補者として石原信雄内閣官房副長官や岩國哲人・島根県出雲市長鳩山邦夫前労相、作家の堺屋太一氏らの名前が浮かんでいるが前提となる政党の枠組みが決まらないため調整は難航。
(Nobuo Ishihara, the deputy chief cabinet secretary, Tetsundo Iwakuni, the mayor of Izumo city in Shimane_Kunio Hatoyama, the former labor minister and the writer Taichi Sakaiya are floated as the candidate_however, since the framework of the predicated political party is not decided_the coordination makes little headway.)

our method:

マルコスと名乗る副司令官が表に出てくるが、実際の司令官は不明。
(While the vice commander_calling himself Marcos appears in public, an actual commander is uncertain.)

baseline:

マルコスと名乗る、_副司令官が表に出てくるが、実際の司令官は不明。
(While the vice commander, calling himself Marcos appears in public, an actual commander is uncertain.)

Figure 4: Comparison of the results of our method and baseline method

ble 7 shows the results of comma insertion at the clause boundaries "topicalized element-*wa*." While there existed 601 commas at such boundaries in the test data, only 141 commas were inserted correctly. We need to consider more effective features about "topicalized element-*wa*."

As for other cases, there existed 130 bunsetsu boundaries between parallel words where commas were not inserted. One example of such case is shown below.

- **correct data:**
  ボウルに豚の背脂、_ニンニク、ショウガ、ネギのみじん切りを入れ、彩りの赤ピーマンも加えます。(Put pork backfat, garlic, ginger and shredded green onion in a bowl, and add red bell peppers for color.)

Table 7: Result of comma insertion at the clause boundaries "topicalized element-*wa*."

recall	precision	F-measure
23.46% (141/601)	59.49% (141/237)	33.65

- **our method:**
  ボウルに豚の背脂ニンニク、ショウガ、ネギのみじん切りを入れ、彩りの赤ピーマンも加えます。(Put pork backfat_ garlic, ginger and shredded green onion in a bowl, and add red bell peppers for color.)

In the correct data, a comma was inserted between the bunsetsu "背脂 (backfat)" and "ニンニク (garlic)."

If a comma should be inserted right after the bunsetsu ''背脂 (backfat)," the number of characters between commas would become too small to be judged as appropriate by the proposed method. So, the feature about the number of characters between commas may have had harmful effects there. On the other hand, a comma was inserted properly between the bunsetsu "ニンニク (garlic)" and "ショウガ (ginger)." This is because *katakana* characters appeared sequentially in addition to appearing as parallel nouns.

6.2 Unnatural Comma Insertion

When commas are inserted at obviously unnatural positions, they have a major impact on the understanding of a sentence by readers. Here, we investigated how many commas had been inserted at obviously unnatural positions by our method. For the article on January 14th (217 sentences, 2,349 bunsetsus) in the test data, we examined 47 commas inserted incorrectly. Three persons decided whether or not the inserted commas were obviously unnatural through consultations. Concretely, when all of the three persons felt that an inserted comma would make readers understand wrongly the meaning of the sentence, the comma was judged to be obviously unnatural.

Among 47 commas, 4 commas were judged obviously unnatural. This result shows that our method is capable of inserting commas at natural positions on some level.

Table 8: Comparison with human judgement

	recall	precision	F-measure
by human	78.30% (249/318)	80.58% (249/309)	79.42
our method	71.07% (226/318)	82.78% (226/273)	76.48

6.3 Comparison with Human Judgement

In our experiment, we evaluated the results of comma insertion of our method by comparing them with the correct data. However, the sufficient level to be reached by automatic comma insertion is uncertain. Here, we evaluated our method by comparing them with the results of comma insertion by another person. By using the same data as used in the subsection 6.2, we conducted an experiment on comma insertion by an annotator who was familiar with writing Japanese documents. Table 8 shows the recall, the precision and the F-measure. The second row shows the results of our method for the same data. As the F-measure of the annotator was 79.42, it turned out that comma insertion task was difficult even for humans. For F-measure, our method achieved 96.30% (76.48/79.42) of the annotator's result. Also, the precision of our method was 82.78%. Although the comma insertion task is difficult, our method was able to properly insert commas.

7 Conclusion

This paper proposed a method for inserting commas into Japanese texts. Our method appropriately inserts commas based on the machine learning method using such features as morphemes, dependencies and clause boundaries. An experiment by using the Kyoto Text Corpus (Kurohashi and Nagao, 1998) showed an F-measure of 75.90, and we confirmed the effectiveness of our method.

The analysis of the experimental results showed that our method cannot insert commas of the particular usage. As a future work, it is necessary to find more useful features for commas of this usage and improve the recall of our method. Also, we will examine "commas emphasizing the adjacent word" which were not included in our targets.

Acknowledgments

This research was partially supported by the Grant-in-Aids for Scientific Research (B) (No. 22300051) and Young Scientists (B) (No. 21700157), and by the Continuation Grants for Young Researchers of The Asahi Glass Foundation.

References

Heidi Christensen, Yoshihiko Gotoh, and Steve Renals. 2001. Punctuation annotation using statistical prosody models. In *Proceedings of ISCA Workshop on Prosody in Speech Recognition and Understanding*, pages 35–40.

Yuqing Guo, Haifeng Wang, and Josef van Genabith. 2010. A linguistically inspired statistical model for Chinese punctuation generation. *ACM Transactions on Asian Language Information Processing*, 9(2):6:1–6:27.

Yoshihiko Hayashi. 1992. A three-level revision model for improving Japanese bad-styled expressions. In *Proceeding of 14th International Conference on Computational Linguistics*, pages 665–671.

Katsuichi Honda. 1982. *Nihongo no sakubun gijutsu (Japanese writing skill)*. Asahi Shimbun Publications Inc. (In Japanese).

Takashi Inukai. 2002. *Moji hyouki tankyuhou (Method of questioning characters and notations)*. Asakura Pulishing Co., Ltd. (In Japanese).

Hideki Kashioka and Takehiko Maruyama. 2004. Segmentation of semantic unit in Japanese monologue. In *Proceedings of International Conference on Speech Language Technology and Oriental COCOSDA*, pages 87–92.

Ji-hwan Kim and P. C. Woodland. 2001. The use of prosody in a combined system for punctuation generation and speech recognition. In *Proceedings of 7th European Conference on Speech Communication and Technology*, pages 2757–2760.

Sadao Kurohashi and Makoto Nagao. 1998. Building a Japanese parsed corpus while improving the parsing system. In *Proceedings of 1st International Conference on Language Resources and Evaluation*, pages 719–724.

Zhang Le. 2008. Maximum entropy modeling toolkit for python and c++. `http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html`. [Online; accessed 1-March-2008].

Yang Liu, Elizabeth Shriberg, Andreas Stolcke, Dustin Hillard, Mari Ostendorf, and Mary Harper. 2006. Enriching speech recognition with automatic detection of sentence boundaries and disfluencies. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1526–1540.

Tohru Shimizu, Satoshi Nakamura, and Tatsuya Kawahara. 2008. Effect of punctuation marks for speech translation unit boundary detection. *IEICE technical report. Speech*, 108(338):127–131. (In Japanese).

Shogakukan's editiorial department. 2007. *kutoten, kigou, hugou katuyoujiten (dictionary of punctuations and symbols)*. Shogakukan. (In Japanese).

Eiji Suzuki, Shizuo Shimada, Kunio Kondo, and Hisashi Sato. 1995. Automatic recognition of optimal punctuation in Japanese documents. In *Proceedings of 50th National Convention of IPSJ*, 50(3):185–186. (In Japanese).

Michael White and Rajakrishnan Rajkumar. 2008. A more precise analysis of punctuation for broad-coverage surface realization with CCG. In *Proceedings of Workshop on Grammar Engineering Across Frameworks*, pages 17–24.

Using Unknown Word Techniques To Learn Known Words

Kostadin Cholakov
University of Groningen
The Netherlands
k.cholakov@rug.nl

Gertjan van Noord
University of Groningen
The Netherlands
g.j.m.van.noord@rug.nl

Abstract

Unknown words are a hindrance to the performance of hand-crafted computational grammars of natural language. However, words with incomplete and incorrect lexical entries pose an even bigger problem because they can be the cause of a parsing failure despite being listed in the lexicon of the grammar. Such lexical entries are hard to detect and even harder to correct.

We employ an error miner to pinpoint words with problematic lexical entries. An automated lexical acquisition technique is then used to learn new entries for those words which allows the grammar to parse previously uncovered sentences successfully.

We test our method on a large-scale grammar of Dutch and a set of sentences for which this grammar fails to produce a parse. The application of the method enables the grammar to cover 83.76% of those sentences with an accuracy of 86.15%.

1 Introduction

In this paper, we present an automated two-phase method for treating *incomplete* or *incorrect* lexical entries in the lexicons of large-scale computational grammars. The performance of our approach is tested in a case study with the wide-coverage Alpino grammar (van Noord, 2006) of Dutch. When applied to real test sentences previously not covered by Alpino, the method causes a parsing coverage of 83.76% and the accuracy of the delivered analyses is 86.15%.

The main advantage of our approach is the successful combination of efficient *error mining* and *lexical acquisition* techniques. In the first phase, error mining pinpoints words which are listed in the lexicon of a given grammar but which nevertheless often lead to a parsing failure. This indicates that the current lexical entry for such a word is either wrong or incomplete and that one or more correct entries for this word are missing from the lexicon. Our idea is to treat the word as if it was unknown and, in the second phase, to employ lexical acquisition (LA) to learn the missing correct entries.

In the case study presented here, we employ the iterative error miner of de Kok et al. (2009). Since it has to be run on a large parsed corpus, we have parsed the Flemish Mediargus corpus (∼1.5 billion words) with Alpino. The reason for this choice is the relatively large lexical difference between standard Dutch and Flemish. This increases the chance to encounter words which are used in Flemish in a way not handled by Alpino yet.

For example, the word *afwater* (to drain) is listed as a first person singular present verb in the Alpino lexicon. However, the error miner identifies this word as the reason for the parsing failure of 9 sentences. A manual examination reveals that the word is used as a neuter noun in these cases– *het afwater* (the drainage). Since there is no noun entry in the lexicon, Alpino was not able to produce full-span analyses.

After the error miner identifies *afwater* as a problematic word, we employ our machine learning based LA method presented in Cholakov and van Noord (2010) to learn new entries for this word. This method has already been successfully applied to the task of learning lexical entries for unknown words and, as the error miner, it can be used 'out of the box'. LA correctly predicts a neuter noun en-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 902–912,
MIT, Massachusetts, USA, 9-11 October 2010.

try for *afwater* and the addition of this entry to the lexicon enables Alpino to cover the 9 problematic sentences from the Mediargus corpus.

It should be noted that since our approach cannot differentiate between incomplete and incorrect entries, no entry in the lexicon is modified. We simply add the lexical entries which, according to the LA method, are most suitable for a given problematic word and assume that, if these entries are correct, the grammar should be able to cover previously unparsable sentences in which the word occurs.

The remainder of the paper is organised as follows. Section 2 describes the error miner. Section 3 presents the Alpino grammar and parser and the LA technique we employ. Section 4 describes an experiment where error mining is performed on the Mediargus corpus and then, LA is applied to learn new lexical entries for problematic words. Section 5 discusses the effect which the addition of the new entries to the lexicon has on the parsing coverage and accuracy. Section 6 provides a comparison between our approach and previous work similar in nature. This section also discusses the application of our method to other systems and languages as well as some ideas for future research.

2 Error Mining

The error miner of de Kok et al. (2009) combines the strengths of the error mining methods of van Noord (2004) and Sagot and de la Clergerie (2006). The idea behind these methods is that grammar errors lead to the parsing failure of some grammatical sentences. By running the grammar over a large corpus, the corpus can be split into two subsets– the set of sentences which received a full-span parse and the set of sentences failed to parse. Words or n-grams which occur in the latter set have a suspicion of being the cause of parsing failures.

van Noord (2004) defines the suspicion of a word sequence as:

(1) $$S(w_i...w_j) = \frac{C(w_i...w_j|error)}{C(w_i...w_j)}$$

where $C(w_i...w_j)$ is the number of sentences which the sequence $w_i...w_j$ occurs in and $C(w_i...w_j|error)$ is the number of occurrences of the sequence in unparsable sentences.

While this method performs well in identifying words and n-grams that are unambiguously suspicious, it also assigns incorrectly a high suspicion to forms which happen to occur often in unparsable sentences by ‘bad luck’. The iterative error mining algorithm of Sagot and de la Clergerie (2006) tackles this problem by taking the following into account:

- If a form occurs within parsable sentences, it becomes less likely for it to be the cause of a parsing failure.
- The suspicion of a form depends on the suspicions of the other forms in the unparsable sentences it occurs in.
- A form observed in a shorter sentence is initially more suspicious than a form observed in a longer one.

However, because of data sparseness problems, this method is only able to handle unigrams and bigrams. Another potential problem is the absence of criteria to determine when to use unigrams and when bigrams to represent forms within a given sentence. Consider the trigram w_1, w_2, w_3 where w_2 is the cause of a parsing failure. In this case, the whole trigram as well as the bigrams w_1, w_2 and w_2, w_3 will become suspicious which would prevent the unigram w_2 from ‘manifesting’ itself.

To avoid this problem, de Kok et al. (2009) uses a preprocessor to the iterative miner of Sagot and de la Clergerie (2006) which iterates through a sentence of unigrams and expands unigrams to longer n-grams when there is evidence that this is useful. A unigram w_1 is expanded to a bigram w_1, w_2 if this bigram is more suspicious than both of its unigrams. The general algorithm is that the expansion to an n-gram $i...j$ is allowed when the following two conditions are fulfilled:

(2) $$\begin{aligned} &S(i...j) > S(i...j-1) \cdot expFactor \\ &S(i...j) > S(i+1...j) \cdot expFactor \end{aligned}$$

Within the preprocessor, suspicion is defined as shown in (1) and the *expFactor* is a parameter specially designed to deal with data sparseness.

As the error mining technique of de Kok et al. (2009) successfully overcomes the problems which the other error mining methods we discussed encounter, we have chosen to employ this technique in our experiment.

3 Automated Lexical Acquisition

3.1 The Alpino Grammar and Parser

Since we employ Alpino for the purposes of our case study, it is convenient to explain the LA method we have chosen to use in the context of this system.

The Alpino wide-coverage parser is based on a large stochastic attribute value grammar. The grammar takes a 'constructional' approach, with rich lexical representations stored in the lexicon and a large number of detailed, construction specific rules (about 800).

Currently, the lexicon contains over 100K lexical entries and a list of about 200K named entities. Each word is assigned one or more lexical types. For example, the verb *amuseert* (to amuse) is assigned two lexical types– *verb(hebben,sg3,intransitive)* and *verb(hebben,sg3,transitive)*– because it can be used either transitively or intransitively. The other type features indicate that it is a present third person singular verb and it forms perfect tense with the auxiliary verb *hebben*.

3.2 Learning Algorithm

The goal of the LA method we describe Cholakov and van Noord (2010) is to assign correct lexical type(s) to a given unknown word.

It takes into account only open-class lexical types: nouns, adjectives and verbs. The types considered in the learning process are called *universal types*[1].

For a given word, a maximum entropy (ME) based classifier takes various morphological and syntactic features as input and outputs a ranked list of lexical types. The probability of a lexical type *t*, given an unknown word and its context *c* is:

$$(3) \qquad p(t|c) = \frac{exp(\sum_i \Theta_i f_i(t,c))}{\sum_{t' \in T} exp(\sum_i \Theta_i f_i(t',c))}$$

where $f_i(t, c)$ may encode arbitrary characteristics of the context and $< \Theta_1, \Theta_2, ... >$ is a weighting parameter which maximises the entropy and can be evaluated by maximising the pseudo-likelihood on a training corpus (Malouf, 2002).

Features
i) a, af, afw, afwa
ii) r, er, ter, ater
iii) particle_yes #in this case *af*
iv) hyphen_no
v) noun⟨het,sg⟩, verb⟨sg1⟩
vi) noun(het,count,sg), noun(de,count,pl)
vii) noun(het), noun(count), noun(sg), noun(de) noun(pl)

Table 1: Features for *afwater*

Table 1 shows the features for *afwater*, the word we discussed in Section 1. Row **(i)** contains 4 separate features derived from the prefix of the word and 4 other suffix features are given in row **(ii)**. The two features in rows **(iii)** and **(iv)** indicate whether the word starts with a particle and if it contains a hyphen, respectively.

Further, the method we describe in Cholakov and van Noord (2009) is applied to generate the paradigm(s) of each word in question. This method uses a finite state morphology to generate possible paradigm(s) for a given word. The morphology does not have access to any additional linguistic information and thus, it generates all possible paradigms allowed by the word orthography. Then, the number of search hits Yahoo returns for each form in a given paradigm is combined with some simple heuristics to determine the correct paradigm(s). The web search heuristics are also able to determine the correct definite article (*de* or *het*) for words with noun paradigms.

One verb and one noun paradigm are generated for *afwater*. In these paradigms, *afwater* is listed as a first person singular present verb form and a singular *het* noun form, respectively. This information is explicitly used as features in the classifier which is shown in row **(v)** of Table 1.

Next, syntactic features for *afwater* are obtained by extracting a number of sentences which it occurs in from large corpora or Internet. These sentences are parsed with a different 'mode' of Alpino where this word is assigned all universal types, i.e. it is treated as being maximally ambiguous. For each sentence only the parse which is considered to be the best by the Alpino statistical disambiguation model

[1] The adjectives can be used as adverbs in Dutch and thus, the latter are not considered to be an open class.

is preserved. Then, the lexical type that has been assigned to *afwater* in this parse is stored. During parsing, Alpino's POS tagger (Prins and van Noord, 2001) keeps filtering implausible type combinations. For example, if a determiner occurs before the unknown word, all verb types are typically not taken into consideration. This heavily reduces the computational overload and makes parsing with universal types computationally feasible.

When all sentences have been parsed, a list can be drawn up with the types that have been used and their frequency:

(4) noun(het,count,sg) 54
noun(de,count,pl) 7
tmp_noun(het,count,sg) 4
adjective(no_e(adv)) 4
proper_name(sg,'ORG') 1

The lexical types assigned to *afwater* in at least 80% of the parses are used as features in the classifier. These are the two features in row **(vi)** of Table 1. Further, as illustrated in row **(vii)**, each attribute of the considered types is also taken as a separate feature.

After the classifier predicts lexical types for each word, these predictions are subject to two additional steps of processing. In the first one, the generated word paradigms are explicitly used as a filtering mechanism. When a word is assigned a verb or an adjective type by the classifier but there is no verb or adjective paradigm generated for it, all verb or adjective predictions for this word are discarded.

The output of this 'filtering' is further processed in the second step which deals with the correct prediction of subcategorization frames for verbs. Following the observations made in Korhonen et al. (2000), Lapata (1999) and Messiant (2008), Cholakov and van Noord (2010) employ a maximum likelihood estimate (MLE) from observed relative frequencies with an empirical threshold to filter out low probability frames.

Since some frames could be very infrequent and the MLE method may not capture them, the generated word paradigms are used to increase the number of contexts observed for a given verb. Additional sentences are extracted for each form in the paradigm of a given word predicted to be a verb. These sentences are again parsed with the universal types. Then we look up the assigned universal verb types, calculate the MLE for each subcategorization frame and filter out frames with MLE below some empirical threshold.

4 Learning New Lexical Entries

Before we start with the description of the experiment, it is important to note that Alpino is very robust– essentially, it always produces a parse. If there is no analysis spanning the whole sentence, the parser finds all parses for each substring and returns what it considers to be the best sequence of non-overlapping parses. However, in the context of this experiment, a sentence will be considered successfully parsed only if it receives a full-span analysis. For the sake of clarity, from now on we shall use the terms *coverage* and *cover* only with regard to such sentences. The term *parsing failure* shall refer to a sentence for which Alpino fails to produce a full-span analysis.

4.1 Error Mining on Mediargus

The first step in our experiment is to perform error mining on the Mediargus corpus. The corpus consists of texts from Flemish newspapers from the period between 1998 and 2007. It contains about 1.5 billion words (∼78M sentences). The corpus has been parsed with Alpino and the parsing results are fed into the error miner of de Kok et al. (2009). The parser has not produced a full-span analysis for 7.28% of the sentences (∼5.7M sentences).

When finished, the error miner stores the results in a data base containing potentially problematic n-grams. Each n-gram is linked to its suspicion score and the sentences which it occurs in and which were not covered by Alpino.

Before proceeding with LA, however, we should identify the n-grams which are indicative for a problem in the lexicon. The first step in this direction is to extract all *unigrams* from the data base which have a suspicion equal to or greater than 0.7 together with the uncovered sentences they occur in. This resulted in a list containing 4179 unique unigrams. Further, we select from this list only those unigrams which have lexical entries in the Alpino lexicon and occur in more than 5 sentences with no full-span parse. Sometimes, the error miner might be wrong about the exact word which causes the parsing failure for a given sentence. The 5 sentences empirical threshold is meant to guarantee that the selected words are systematically causing problems for the parser.

The result of this selection is 36 unigrams (words) which occur in a total of 388 uncovered sentences– an average of 10.78 sentences per word. The small number of selected words is due to the fact that most of the problematic 4179 unigrams represent tokenization errors (two or more words written as one) and spelling mistakes which, naturally, are not listed in the Alpino lexicon. Very few of the 4179 unigrams are actual unknown words. Table 2 shows some of the problematic unigrams and their suspicions.

opVorig	0.898989
GentHoewel	0.89759
Nieuwpoortl	0.897414
SportTijdens	0.897016
DirvenDe	0.896428
mistrap	0.896038
Dwoeurp	0.896013
passerde	0.89568
doorHugo	0.893901
goedkmaken	0.892407
ManneN	0.891539
toegnag	0.891523

Table 2: Problematic unigrams and their suspicions

It can be seen immediately that most of the unigrams presented in the table are tokenization errors. There are also some typos. The unigram *passerde* should be written as *passe**e**rde*, the past singular verb form of the verb ‘to pass’ and *toegnag* is the misspelled noun *toegang* (access). The only problematic unigram with a lexical entry in the Alpino lexicon is *mistrap* (misstep, to misstep).

Although the experiment setup yields a small test set, we employ it because the words in this set represent ‘clear-cut’ cases. This allows us to demonstrate better the effect of our technique.

4.2 Applying Lexical Acquisition

Our assumption is that incomplete or incorrect lexical entries prevented the production of full-span parses for the 388 sentences in which the 36 problematic words pinpointed by the error miner occur. That is why, in the second step of the experiment, these words are temporarily removed from the Alpino lexicon, i.e. they are treated as unknown words, and we employ the LA method presented in the previous section to learn offline new lexical entries for them.

The setup for the learning process is exactly the same as in Cholakov and van Noord (2010). The set of universal types consists of 611 types and the ME-based classifier has been trained on the same set of 2000 words as in Cholakov and van Noord (2010). Those types predicted by the classifier which account together for less than 5% of probability mass are discarded.

In order to increase the number of observed contexts for a given word when parsing with the universal types, up to 100 additional sentences in which the word occurs are extracted from Internet. However, when predicting new lexical entries for this word, we want to take into account only sentences where it causes a parsing failure. It is in such sentences where a new lexical entry can be learnt through LA. For example, the LA method would be able to predict a noun entry for *afwater* if it focuses only on contexts where it has a noun reading, i.e. on sentences not covered by Alpino.

That is why, the sentences we extracted from Internet are first parsed with the standard Alpino configuration. When averaging over the 36 sentence sets, it turns out that Alpino has been able to cover 10.05% of the sentences. Although we cannot be sure that the 36 words are the cause of a parsing failure in each of the uncovered sentences, this low coverage indicates once more that Alpino has systematic problems with sentences containing these words.

Then, the uncovered sentences from Internet together with the 388 problematic sentences from the Mediargus corpus are parsed with Alpino and the universal types. For example, the list of universal types assigned to *afwater* in (4) contains mostly noun types, i.e. the kind of types which are currently not in the lexicon for this word and which we want to learn.

The result of the LA process is the prediction of a total of 102 lexical types, or 2.83 types per word. This high number is due to the fact that 25 words receive verb predictions. Since a verb can have various subcategorization frames, there is one type assigned for each frame. For example, *inscheppen* (to spoon in(to)) receives 3 types which differ only in the subcategorization frame– *verb(hebben,inf,tr.)*, *verb(hebben,inf,intr.)* and *verb(hebben,inf,np_np)*. However, the infinitive in Dutch is also the form for plural present and *inscheppen* correctly receives 3 more predictions– *verb(hebben,pl,tr.)*, *verb(hebben,pl,intr.)* and *verb(hebben,pl,np_np)*.

Let us examine the most frequent types of lexicon errors for the 36 problematic words by looking at the current Alpino lexical entries for some of these words and the predictions they receive from the LA method. The original Alpino entries for 19 of the 25 words predicted to be verbs are a product of a specific lexical rule in the grammar. Consider the following sentences:

(5) a. Ik schep de soep in de kom.
I spoon the soup in the bowl
'I spoon the soup into the bowl.'
b. dat ik de soep de kom in schep
that I the soup the bowl in spoon
'that I spoon the soup into the bowl'
c. dat ik de soep de kom *in*schep
that I the soup the bowl in spoon
'that I spoon the soup into the bowl'

We see in (5-b) that the preposition *in* is used as a postposition in the relative clause. However, in such cases, there is linguistic evidence that *in* behaves as a separate verb particle. That is why, as shown in (5-c), people sometimes write *in* and the verb together when they occur next to each other in the sentence. To account for this, Alpino employs a special lexical rule. This rule assigns a certain type of subcategorization frame to verbs like *inscheppen* where a postposition can be interpreted as a separable particle. That subcategorization frame requires a noun phrase ('the soup' in (5-c)) and a locative NP ('the bowl' in (5-c)).

However, in some cases, the entries generated by this lexical rule cannot account for other possible usages of the verbs in question. For example,

(6) U moet deze zelf inscheppen.
you must this yourself spoon in.INF
'You have to spoon this in yourself.'

Alpino fails to parse this sentence because *inscheppen* is used without a locative NP. Now, when the LA method has predicted a transitive verb type for *inscheppen*, the parser should be able to cover the sentence. Other such examples from our data include ***weg****wist* (to erase.3PER.SG), ***onder****ligt* (to lie under.3PER.SG), etc.

Further, there are 10 words, including *afwater*, which represent cases of nominalisation currently not accounted for in the Alpino lexicon. The LA process correctly predicts noun types for these words. This should enable the parser to cover sentences like:

(7) Die moet een deel van het afwater vervoeren.
this must a part from the drainage transport/move
'This has to move a part of the drainage.'

where *afwater* is used as a noun.

There are also 3 words which correctly receive adjective predictions. Currently, their lexical entries are incomplete because they are assigned only past participle types in the lexicon. However, past participles in Dutch can also act as adjectives. For historical reasons, this systematic ambiguity is not treated as such in Alpino. Each participle should also have a separate adjective lexical entry but, as we see, this is not always the case.

5 Results

After LA is finished, we restore the original lexical entries for the 36 words but, additionally, each word is also assigned the types which have been predicted for it by the LA method. The 388 problematic sentences from the Mediargus corpus are then re-parsed with Alpino. We are interested in observing:

1. how many sentences receive a full-span analysis

2. how the parsing accuracy of Alpino changes

Table 3 shows that when the Alpino lexicon is extended with the lexical entries we learnt through LA, the parser is able to cover nearly 84% of the sentences, including the ones given in (6) and (7). Since there is no suitable baseline which this result can be compared to, we developed an additional model which indicates what is likely to be the maximum coverage that Alpino can achieve for those sentences by adding new lexical entries only.

In this second model, for each of the 36 words, we add to the lexicon all types which were successfully used for the respective word during the parsing with universal types. In this way, Alpino is free to choose from all types it has considered suitable for a given word, i.e. the parser is not limited by the outcome of the LA process but rather by the overall quality of the grammar.

The 'universal types' model performs better than ours– it achieves 87.9% coverage. Still, the performance of our model is close to this result, i.e. close to what we consider to be the maximal possible coverage of Alpino for these 388 sentences when only LA is used.

Model	Coverage (%)
Our model (Alpino + LA)	83.76
Universal types	87.89

Table 3: Coverage results for the re-parsed 388 problematic sentences

Some of the sentences which cannot be covered by both models are actually not proper sentences but fragments which were wrongly identified as sentences during tokenization. Many other cases include sentences like:

(8) Een gele frommel papier, Arabische lettertekens.
a yellow crease paper Arabic characters
'A yellow paper crease, Arabic characters.'

which is probably the caption of a photo or an illustration. However, because of the absence of a verb, Alpino splits the analysis into two parts– the part before the comma and the part after the comma.

Here is a more interesting case:

(9) Als we ons naar de buffettafel begeven, mistrap ik me.
when we us to the buffet proceed misstep I myself
'When we proceed to the buffet I misstep.'

The LA method does not predict a reflexive verb type for *mistrap* which prevents the production of a full-span analysis because Alpino cannot connect the reflexive pronoun *me* to *mistrap*. In this case, however, the universal type model outperforms ours. A reflexive verb type is among the universal types and thus, Alpino is able to use that type to deliver a full-span parse. We should note though, that LA correctly predicts a noun type for *mistrap* which enables Alpino to parse successfully the other 14 sentences which this word occurs in.

Let us now look at the correctness of the delivered parses. To estimate the accuracy of the parser, we have randomly selected 100 sentences out of the 388 sentences in the test set and we have manually annotated them in order to create a gold standard for evaluation.

Accuracy in Alpino is measured in terms of dependency relations. The accuracy for sentences which are not assigned a full-span analysis but a sequence of non-overlapping parses can still be larger than zero because, within these parses, some correct dependency relations could have been produced. That is why, though the coverage of Alpino for the selected 100 sentences is zero, we can still obtain a number for accuracy and use it as a baseline for comparison. Clearly, this baseline is expected to perform worse than both our model and the universal types one since those are able to cover most of the sentences and thus, they are likely to produce more correct dependency relations. However, it gives us an idea how much extra quality is gained when coverage improves.

The accuracy results for the 100 annotated sentences are given in Table 4. The average sentence length is 18.9 tokens.

Model	Accuracy (%)	msec/sentence
Alpino	63.35	803
Our model	86.15	718
Universal types	85.12	721

Table 4: Accuracy results for the 100 annotated sentences

Our model achieves the highest accuracy without increasing the parse times. Further, the baseline has a much lower result which shows that coverage is not gained on the expense of accuracy.

Our model and the universal types one achieve the same accuracy for most of the sentences. However, the universal types model has an important disadvantage which, in some cases, leads to the production of wrong dependency relations. The model predicts a large number of lexical types which, in turn, leads to large *lexical ambiguity*. This lexical ambiguity increases the number of possible analyses Alpino chooses from, thus making it harder for the parser to produce the correct analysis. Let us consider the following example where a sentence is covered by both models but the universal types model has lower accuracy:

(10) Dat wij het rechttrokken, pleit voor onze
that we it straighten.PAST.PL. plead for our
huidige conditie.
current condition
'It pleads for our condition that we straightened it.'

Here, *het* is the object of the verb *rechttrokken*. However, although there are transitive verb types among the universal types assigned to *rechttrokken*, Alpino chooses to use a verb type which subcategorizes for a measure NP. This causes for *het* to be analysed not as an object but as a measure complement, i.e. the produced dependency relation is incorrect.

The LA method, on the other hand, is much more restrictive but its predictions are also much more accurate. Since it considers sentences containing other forms of the paradigm of *rechttrokken* when predicting subcategorization frames, the LA method correctly assigns only one transitive and one intransitive verb type to this word. This allows Alpino to recognize *het* as the object of the verb and to produce the correct dependency relation.

The few cases where the universal types model outperforms ours include sentences like the one given in (9) where the application of our model could not enable Alpino to assign a full-span analysis. Sometimes, the LA method is too restrictive and does not output some of the correct types. These types, on the other hand, could be provided by the universal types model and could enable Alpino to cover a given sentence and thus, to produce more correct dependency relations. Allowing for the LA method to predict more types, however, has proven to be a bad solution because, due to the increased lexical ambiguity, this leads to lower parsing accuracy.

6 Discussion

6.1 Comparison to Previous Work

The performance of the technique we presented in this paper can be compared to the performance of a number of other approaches applied to similar tasks.

Zhang et al. (2006) and Villavicencio et al. (2007) use error mining to semi-automatically detect English multiword expressions (MWEs). Then, they employ LA to learn proper lexical entries for these MWEs and add them to the lexicon of a large-scale HPSG grammar of English (ERG; (Copestake and Flickinger, 2000)). This increases parsing coverage by 15% to 22.7% for a test set of 674 sentences containing MWEs and parsed with the PET parser (Callmeier, 2000). In both studies, however, the combination of error mining and LA is applied to a very specific task whereas our method is a general one.

Nicolas et al. (2008) employ a semi-automatic method to improve a large-scale morphosyntactic lexicon of French (Le*fff*; (Sagot et al., 2006)). The lexicon is used in two grammars– the FRMG (Thomasset and de la Clergerie, 2005), a hybrid Tree Adjoining/Tree Insertion Grammar, and the SxLFG-FR LFG grammar (Boullier and Sagot, 2006). The first step in this approach is also the application of an error miner (Sagot and de la Clergerie, 2006) which uses a parsed newspaper corpus (about 4.3M words) to pinpoint problematic unigrams.

The crucial difference with our method is in the second step. Nicolas et al. (2008) assign *underspecified* lexical entries to a given problematic unigram to allow the grammar to parse the uncovered sentences associated with this unigram. Then, these entries are ranked based on the number of successful parses they have been used in.

The use of underspecification, however, causes large ambiguity and severe parse overgeneration (observed also in Fouvry (2003)). As a consequence of that, the ranked list of lexical entries for each unigram is manually validated to filter out the wrong entries. The employment of LA in our approach, on the other hand, makes it fully automatic. The ranking of the predictions is done by the classifier and the predicted entries are good enough to improve the parsing coverage and accuracy without any manual work involved. Generally, recent studies (Baldwin, 2005; Zhang and Kordoni, 2006; Cholakov et al., 2008; Cholakov and van Noord, 2010) have clearly shown that when it comes to learning new lexical entries, elaborate LA techniques perform better and are more suitable for large-scale grammars than underspecification[2].

Further, the naive ranking system used in Nicolas et al. (2008) puts a correctly generated entry for an infrequent usage of a given word (e.g., a verb with a rare subcat frame) in the bottom of the ranked list because of the low number of sentences in which this entry is used. The LA method we employ is more sensitive to rare usages of words because it considers occurrences of the word in question outside the parsed corpus (very important if the corpus is domain-specific) and it also takes into account all forms in the paradigm(s) of the word. This increases the chances of a rare usage of this word to 'manifest' itself.

Nicolas et al. (2008) uses the lexical entries which remain after the manual validation to re-parse the newspaper corpus. 254 words (mostly verbs) are corrected and the parse coverage increases by 3.4% and 1.7% for the FRMG and the SxLFG, respectively. However, the authors do not mention how many of the original uncovered sentences they are able to cover and therefore, we cannot compare our coverage result. Nothing is said about the parsing accuracy. Even with manually validated lexical entries, it is still possible for the grammar to produce full-span but wrong analyses.

6.2 Application to Other Systems and Languages

It is important to note that this paper should be viewed as a case study where we illustrate the results of the application of what we believe to be a good algorithm for dealing with incomplete or incorrect lexical entries– namely, the combination of error mining and LA. However, our method is general enough to be applied to other large-scale grammars and languages.

The error mining is directly usable as soon as there is a large parsed corpus available. The LA technique we employed is also quite general provided that certain requirements are fulfilled. First, words have to be mapped onto some finite set of labels of which a subset of open-class (universal) labels has to be selected. This subset represents the labels which can be predicted for unknown words. Second, we need a parser to analyse sentences in which a given unknown word occurs. Finally, the ME-based classifier allows for arbitrary combinations of features and therefore, any (language-specific) features considered useful can be included. As for the paradigm generation method, the idea of combining a finite state morphology and web heuristics is general enough to be implemented for different languages.

We have already started investigating the applicability of our method to the FRMG and a large-scale grammar of German and the initial experiment and results we have obtained are promising.

6.3 Future Research

Currently, our algorithm handles only unigrams (words). However, it would be useful to extend it, so it can work with longer n-grams. For example, a given word could have some reading which is not yet handled in the lexicon only within a particular bi- or trigram.

Consider the bigram '*schampte af*' which has been identified as problematic by the error miner. It represents the particle verb '*afschampte*' (to glance.PAST.SG). Although the lexicon contains a verb entry for '*schampte*', there is no entry handling the case when this verb combines with the particle '*af*'. Another example is the bigram '*de slachtoffer*' (the victim). In standard Dutch, the noun '*slachtoffer*' goes with the '*het*' definite article which is marked in its lexical entry. However, in Flemish it is used with the '*de*' article.

Our method is currently not able to capture these two cases since they can be identified as problematic on bigram level and not when only unigrams are considered.

Further, the definition of what the error miner considers to be a successful parse is a rather crude one. As we saw, even if the grammar is able to produce a full-span analysis for a given sentence, this analysis could still not be the correct one. Therefore, it is possible that a word could have a problematic lexical entry even if it only occurs in sentences which are assigned a full-span parse. Currently, such a word will not be identified as problematic by the error miner. That is why, some (statistical) model which is capable of judging the plausibility of a parse should be developed and incorpo-

[2] In Nicolas et al. (2008) the authors also admit that an elaborate LA technique will produce better results.

rated in the calculation of the suspicions during error mining.

References

Tim Baldwin. 2005. Bootstrapping deep lexical resources: Resources for courses. In *Proceedings of the ACL-SIGLEX 2005 Workshop on Deep Lexical Acquisition*, Ann Arbor, USA.

Pierre Boullier and Benoît Sagot. 2006. Efficient parsing of large corpora with a deep LFG parser. In *Proceedings of LREC'06*, Genoa, Italy.

Ulrich Callmeier. 2000. PET– a platform for experimentation with efficient HPSG processing techniques. In *Journal of Natural Language Engineering*, volume 6, pages 99–107. Cambridge University Press.

Kostadin Cholakov and Gertjan van Noord. 2009. Combining finite state and corpus-based techniques for unknown word prediction. In *Proceedings of the 7th Recent Advances in Natural Language Processing (RANLP) conference*, Borovets, Bulgaria.

Kostadin Cholakov and Gertjan van Noord. 2010. Acquisition of unknown word paradigms for large-scale grammars. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING-2010)*, Beijing, China.

Kostadin Cholakov, Valia Kordoni, and Yi Zhang. 2008. Towards domain-independent deep linguistic processing: Ensuring portability and re-usability of lexicalised grammars. In *Proceedings of COLING 2008 Workshop on Grammar Engineering Across Frameworks (GEAF08)*, Manchester, UK.

Ann Copestake and Dan Flickinger. 2000. An open-source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of the 2nd International Conference on Language Resource and Evaluation (LREC 2000)*, Athens, Greece.

Daniël de Kok, Jianqiang Ma, and Gertjan van Noord. 2009. A generalized method for iterative error mining in parsing results. In *Proceedigns of the 2009 Workshop on Grammar Engineering Across Frameworks, ACL-IJCNLP 2009*, pages 71–79, Singapore.

Frederik Fouvry. 2003. Lexicon acquisition with a large-coverage unification-based grammar. In *Companion to the 10th Conference of EACL*, pages 87–90, Budapest, Hungary.

Anna Korhonen, Genevieve Gorell, and Diana McCarthy. 2000. Statistical filtering and subcategorization frame acquisition. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, Hong Kong, China.

Mirella Lapata. 1999. Acquiring lexical generalizations from corpora. A case study for diathesis alternations. In *Proceedings of the 37th Annual Meeting of ACL*, Maryland, USA.

Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th conference on Natural Language Learning (CoNLL-2002)*, pages 49–55, Taipei, Taiwan.

Cedric Messiant. 2008. A subcategorization acquisition system for French verbs. In *Proceedings of the ACL 2008 Student Research Workshop*, Columbus, OH.

Lionel Nicolas, Benoît Sagot, Miguel Molinero, Jacques Farré, and Eric de la Clergerie. 2008. Computer aided correction and extension of a syntactic wide-coverage lexicon. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-2008)*, pages 633–640, Manchester, UK.

Robbert Prins and Gertjan van Noord. 2001. Unsupervised POS-tagging improves parcing accuracy and parsing efficiency. In *Proceedings of IWPT*, Beijing, China.

Benoît Sagot and Eric de la Clergerie. 2006. Error mining in parsing results. In *Proceedings of the 44th Meeting of the Association for Computational Linguistics (ACL'06)*, pages 329–336, Morristown, NJ, USA.

Benoît Sagot, Lionel Clément, Eric de la Clergerie, and Pierre Boullier. 2006. The Lefff 2 syntactic lexicon for French. In *Proceedings of LREC'06*, Genoa, Italy.

François Thomasset and Eric de la Clergerie. 2005. Comment obtenir plus des méetagrammaires. In *Proceedings of TALN'05*, Dourdan, France.

Gertjan van Noord. 2004. Error mining for wide-coverage grammar engineering. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04)*, pages 446–453, Barcelona, Spain.

Gertjan van Noord. 2006. At last parsing is now operational. In *Proceedings of TALN*, Leuven, Belgium.

Aline Villavicencio, Valia Kordoni, Yi Zhang, Marco Idiart, and Carlos Ramisch. 2007. Validation and evaluation of automatically acquired multiword expressions for grammar engineering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1034–1043, Prague, Czech Republic.

Yi Zhang and Valia Kordoni. 2006. Automated deep lexical acquisition for robust open text processing. In *Proceedings of the Fifth International Conference on Language Resourses and Evaluation (LREC 2006)*, Genoa, Italy.

Yi Zhang, Valia Kordoni, Aline Villavicencio, and Marco Idiart. 2006. Automated multiword expression prediction for grammar engineering. In *Proceedings of* *the ACL Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, pages 36–44, Sydney, Australia.

WikiWars: A New Corpus for Research on Temporal Expressions

Paweł Mazur[1,2]

[1]Institute of Applied Informatics,
Wrocław University of Technology
Wyb. Wyspiańskiego 27,
50-370 Wrocław, Poland
pawel@mazur.wroclaw.pl

Robert Dale[2]

[2]Centre for Language Technology,
Macquarie University,
NSW 2109, Sydney, Australia
Pawel.Mazur@mq.edu.au
Robert.Dale@mq.edu.au

Abstract

The reliable extraction of knowledge from text requires an appropriate treatment of the time at which reported events take place. Unfortunately, there are very few annotated data sets that support the development of techniques for event time-stamping and tracking the progression of time through a narrative. In this paper, we present a new corpus of temporally-rich documents sourced from English Wikipedia, which we have annotated with TIMEX2 tags. The corpus contains around 120000 tokens, and 2600 TIMEX2 expressions, thus comparing favourably in size to other existing corpora used in these areas. We describe the preparation of the corpus, and compare the profile of the data with other existing temporally annotated corpora. We also report the results obtained when we use DANTE, our temporal expression tagger, to process this corpus, and point to where further work is required. The corpus is publicly available for research purposes.

1 Introduction

The reliable processing of temporal information is an important step in many NLP applications, such as information extraction, question answering, and document summarisation. Consequently, the tasks of identifying and assigning values to temporal expressions have recently received significant attention, resulting in the creation of mature corpus annotation guidelines (e.g. TIMEX2[1] and TimeML[2]), publicly available annotated corpora (ACE,[3] TimeBank[4]) and a number of automatic taggers (see, for example, (Mani and Wilson, 2000; Schilder, 2004; Hacioglu et al., 2005; Negri and Marseglia, 2005; Saquete, 2005; Han et al., 2006; Ahn et al., 2007)).

However, existing corpora have their limitations. In particular, the documents in these corpora tend to be limited in length and, in consequence, discourse structure. This impacts on the number, range and variety of temporal expressions they contain. Existing research carried out on the interpretation of temporal expressions, e.g. by (Baldwin, 2002; Ahn et al., 2005; Mazur and Dale, 2008), suggests that many temporal expressions in documents, especially news stories, can be interpreted fairly simply as being relative to a reference date that is typically the document creation date. This phenomenon does not carry over to longer, more narrative-style documents that describe extended sequences of events, as found, for example, in biographies or descriptions of protracted geo-political events. Consequently, existing corpora are not ideal as development data for systems intended to work on such historical narrations.

In this paper we introduce a new annotated corpus of temporal expressions that is intended to address this shortfall. The corpus, which we call WikiWars, consists of 22 documents from English Wikipedia that describe the historical course of wars. Despite the small number of documents, their length means that the corpus yields a large number of temporal expressions, and poses new challenges for tracking

[1] See http://fofoca.mitre.org.

[2] See http://timeml.org.

[3] See corpora LDC2005T07 and LDC2006T06 in the LDC catalogue (http://www.ldc.upenn.edu).

[4] See corpus LDC2006T08 in the LDC catalogue.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 913–922,
MIT, Massachusetts, USA, 9-11 October 2010.

temporal focus through extended texts. The corpus has been made available for others to use;[5] to give an indication of the difficulty of processing the temporal phenomena in the texts, we also report on the performance of DANTE, our temporal expression tagger, on detecting and interpreting the temporal expressions in the corpus.

The rest of this paper is organised as follows. In Section 2 we describe related work, focusing on the TIMEX2 annotation scheme, and existing corpora that contain annotations of temporal expressions using this scheme. Section 3 describes the process of creation of the WikiWars corpus. In Section 4 we comment on some artefacts of Wikipedia articles that impact on the annotation process and the use of this corpus. Then, in Section 5 we analyse the differences between the WikiWars corpus and the widely-used ACE corpora. In Section 6 we report on the performance of our temporal expression tagger on this data set. Finally, in Section 7, we conclude.

2 Related Work

At the time of writing, there are two mature, wide-coverage schemes for the annotation of temporal information in texts: TIMEX2 (Ferro et al., 2005) and TimeML (Pustejovsky et al., 2003; Boguraev et al., 2005), which is soon to become an ISO standard (Pustejovsky et al., 2010).

These schemes were used to annotate corpora that are often used in research on temporal expression recognition and normalisation: the series of corpora used for training and evaluation in the Automatic Content Extraction (ACE) program[6] run in 2004, 2005 and 2007, and the TimeBank Corpus.

The ACE corpora were prepared for the development and evaluation of systems participating in the ACE program. However, the evaluation corpora have never been publicly released, and thus are currently, for all practical purposes, unavailable. The ACE 2004 corpus contains news data only (broadcast news, newspaper and newswire), while the ACE 2005 and 2007 corpora contain news (broadcast and newswire), conversations (broadcast and telephone), UseNet discussions and web blogs. The 2005 and 2007 ACE corpora are annotated with the latest version of TIMEX2 (2005), while the 2004 corpus is annotated with the older 2003 version of TIMEX2; however, the differences are not very significant.

Apart from the unavailability of the evaluation data, there are two issues with the ACE corpora. One is that most of the documents are relatively short, so that the average number of temporal expressions per document is low (typically between seven and nine per document, including the document time stamp as a metadata element). This results in very limited temporal discourse structure, and relatively few underspecified and relative temporal expressions. Unfortunately, these are the more difficult temporal expressions to handle, and so the ACE corpora may not serve as a good baseline for performance more generally.

A second problem is that the ACE corpora appear to contain a significant number of errors in the gold standard annotations, with respect to both the annotated extents and the semantic values assigned, which do not always follow the TIMEX2 guidelines.

TimeBank v1.2 is a revised and improved version of TimeBank 1.1 resulting in a number of errors fixed and inconsistencies removed (see (Boguraev et al., 2007)). Unfortunely, this corpus has the same limitations as the ACE corpora in regard to document length and complexity of discourse structure. Further, TimeBank is annotated with TimeML, a scheme more complex than TIMEX2 since it also encompasses the tagging of events and temporal relations. However, TIMEX2 is sufficiently sophisticated for the annotation of most types of temporal expressions, and our review of the literature reveals that the majority of existing temporal taggers output TIMEX2 annotations. Since automatic conversion between TIMEX2 and TimeML annotations is not straightforward, TimeBank is of limited use for those who work specifically with TIMEX2.

3 Creating WikiWars

Given the above concerns, we were particularly interested in developing a corpus that would allow more rigorous testing of techniques for tracking time across extended narratives, since these give rise to more complex temporal phenomena than are found in simpler documents. To avoid copyright issues that might arise in the development and distribution of such a

[5] See www.TimexPortal.info/WikiWars.

[6] See www.itl.nist.gov/iad/mig/tests/ace.

corpus, we decided to use Wikipedia as a source. After considering various types of historical narrative, we settled on descriptions of the course of wars and conflicts as being particularly rich in the kinds of phenomena we wanted to explore.

3.1 Selecting Data

We queried Google with two phrases, 'most famous wars in history' and 'the biggest wars', and in each case chose the top-ranked result. One of the pages found proposed a list of the 10 most famous wars in history, and the other listed the names of the 20 biggest wars that happened in the 20th century, measured in terms of the number of military deaths. We combined the two lists, eliminated duplicates, and searched Wikipedia for articles describing these wars. Wikipedia did not contain an article for one war, and we considered two articles as inappropriate for our purposes since they did not describe the course of the wars, but rather some general information about the conflicts. This resulted in a final set of 22 articles. More details of the selection process and the URLs of the chosen Wikipedia articles are provided in the documentation distributed with the corpus.

3.2 Text Extraction and Preprocessing

To prepare the corpus, we first manually copied text from those sections of the webpages that described the course of the wars. This involved manual removal of picture captions and cross-page links. We then ran a script over the results of this extraction process to convert some Unicode characters into ASCII (ligatures, spaces, apostrophes, hyphens and other punctuation marks), and to remove citation links and a variety of other Wikipedia annotations.

Finally, we converted each of the text files into an SGML file: each document was wrapped in one `DOC` tag, inside which there are `DOCID`, `DOCTYPE` and `DATETIME` tags. The document time stamp is the date and time at which we downloaded the page from Wikipedia to our local repository. The proper content of the article is wrapped in a `TEXT` tag. This document structure intentionally follows that of the ACE 2005 and 2007 documents, so as to make the processing and evaluation of the WikiWars data highly compatible with the tools used to process the ACE corpora.

3.3 Creating Gold Standard Annotations

Having prepared the input SGML documents, we then processed them with the DANTE temporal expression tagger (see Mazur and Dale (2007)). DANTE outputs the original SGML documents augmented with an inline TIMEX2 annotation for each temporal expression found. These output files can be imported to Callisto,[7] an annotation tool that supports TIMEX2 annotations. Using a temporal expression tagger as a first-pass annotation tool not only significantly reduces the amount of human annotation effort required (creating a tag from scratch requires a number of clicks in the annotation tool), but also helps to minimize the number of errors that arise from overlooking markable expressions through 'annotator blindness'. The annotations produced by DANTE were then manually corrected in Callisto via the following process. First, Annotator 1 (the first author) corrected all the annotations produced by DANTE, both in terms of extent and the values provided for TIMEX2 attributes. This process also included the annotation of any temporal expression missed by the automatic tagger, and the removal of spurious matches. Then, Annotator 2 (the second author) checked all the revised annotations and prepared a list of errors found and doubts or queries in regard to potentially problematic annotations. Annotator 1 then verified and fixed the errors, after discussion in the case of disagreements.

The final SGML files containing inline annotations were then transformed into ACE APF XML annotation files, this being the stand-off markup format developed for ACE evaluations. This transformation was carried out using the `tern2apf` tool developed by NIST for the ACE 2004 evaluations, with some modifications introduced by us to adjust the tool to support ACE 2005 documents and to add a document ID as part of the ID of a TIMEX2 annotation (so that all annotations would have corpus-wide unique IDs).

The resulting corpus is thus available in two formats: one contains the original documents enriched with inline annotations, and the other consists of stand-off annotations in the ACE APF format.

[7] See `http://callisto.mitre.org`.

3.4 Some Deficiencies of TIMEX2

The annotation process described above revealed some issues with the use of TIMEX2 in practice. First, the flexibility of the TIMEX2 scheme, which can be at first seen as an advantage, actually makes it ambiguous. One instance of this phenomenon relates to the fact that the TIMEX2 guidelines state that the provision of some attribute values for what are called **event-based expressions** (such as *three weeks after the siege of Boston began* or *the first year of the American invasion*) is optional. Since our corpus has a significant number of such expressions, the decision as to whether or not to provide semantic values in such cases has a potentially large impact on the perceived performance of a tagger. In such cases, we decided only to provide the value when it is very clear from the article itself what the value should be.

Another area where TIMEX2 is not ideal is in regard to the annotation of time zones. First, only whole-hour time differences are supported, which eliminates some time zones (e.g. Afghanistan lies in UTC+04:30). Second, time zone information is supposed to be marked only for expressions which have it explicitly stated. However, it can often be inferred from the context that subsequent unadorned time references should inherit the same time zone as an earlier time reference.

We also found that, in a not insignificant number of cases, it is impossible to provide a precise and correct value for a temporal expression. For example, the TIMEX2 guidelines stipulate that the anchors of durations cannot have a MOD attribute, so if the anchor is *mid-August*, the value of the anchor must refer to August, which is not entirely correct as the semantics of *mid-* is lost.

TIMEX2 only supports nonspecific expressions which have explicit information about granularity. Expressions such as *a very short time* or *a short period of time* therefore cannot be provided with any value, since the context does not indicate whether the period involved should be measured in days, weeks, or months. One might consider using the typical durations of events of the corresponding types in such cases, but this solution also has problems (see (Pan et al., 2006)).

As is acknowledged in the TIMEX2 guidelines, the treatment of set expressions (i.e. recurring times and durations and frequencies, e.g. *twice a month*) is underdeveloped. One rule states that set expressions should not be anchored (Ferro et al., 2005, p. 42); this has the consequence that the full semantics of the expression *annually since 1955* cannot be provided, and the expression is therefore treated as two separate expressions, *annually* and *1955*.

Finally, alternative calendars are not supported, so an expression like *February in the pre-revolutionary Russian calendar* cannot receive a value unless it appears in an appositive construction which provides an alternative description. Similarly, consider Example (1):

(1) On *9 November 1799* (18 Brumaire of *the Year VIII*) Napoleon Bonaparte staged the coup of 18 Brumaire which installed the Consulate.

Here, *18 Brumaire of the Year VIII* is a date in an alternative calendar used in France, but we annotated only *the Year VIII* based on the trigger *year*. Note that *18 Brumaire* also occurs later in the sentence, but is not annotated.

3.5 Corpus Statistics

The corpus contains 22 documents with a total of almost 120,000 tokens[8] and 2,671 temporal expressions annotated in TIMEX2 format. In Table 1 we compare the WikiWars corpus with the other existing corpora. While the ACE 2005 Training corpus remains the largest corpus, WikiWars is larger than the ACE 2005 and 2007 evaluation corpora and the TimeBank v1.2 corpus, both in terms of number of tokens and TIMEX2 annotations. WikiWars has an order of magnitude more temporal expressions in each document, and a slightly higher density of temporal expressions than the other corpora.

Table 2 presents statistics on the individual documents that make up the corpus. The documents vary considerably in size, the smallest consisting of only 1,455 tokens, and the largest being eight times larger at 11,640 tokens. The density of TIMEX2 annotations varies from 1 in 23.1 tokens to 1 in 72.1 tokens, but for the majority of documents the ratio lies between 30 and 60.

[8] All token counts presented in Tables 1 and 2 were obtained using GATE's default English tokeniser; hyphenated words, e.g. *British-held* and *co-operation*, were treated as single tokens. For more information on GATE see (Cunningham et al., 2002).

Corpus	Docs	KB	Tokens	Temp. Expr.	$\frac{\text{Tokens}}{\text{TIMEX}}$	$\frac{\text{TIMEX}}{\text{Doc}}$
ACE05 Train.	599	1,733	318,785	5,469	58.3	9.13
ACE05 Eval.	155	350	63,217	1,154	54.8	7.45
ACE07 Eval.	254	561	104,779	2,028	51.7	7.98
WikiWars	22	631	119,468	2,671	44.7	121.41
TimeBank1.2	183	816	78,444	1,414	55.5	7.73

Table 1: Statistics of the Wikipedia War corpus compared to those of other corpora.

4 The Nature of Wikipedia Articles

Wikipedia articles may be edited by a large number of people over a significant number of revisions. We checked how often the articles constituting WikiWars were modified in the period from January 2008 to February 2010. On average, each article was changed almost 52 times per month, with the monthly number of changes for a single article ranging from 1 to 372.[9] The minimum average for an individual document was 13.08 (17_AlgerianWar), and the maximum was 171.77 (07_IraqWar).

The nature of the revision process in Wikipedia leads to some artefacts that may be not typical of other document sources, such as news, where the text is usually carefully prepared by its author and checked by an editor. This is not to say that Wikipedia content is necessarily of low quality; this is an encyclopedia with many people and bots controlling its quality, and there exist manuals of style for authors to help them avoid errors and ambiguity and to ensure maximum consistency.[10] However, given the large number of editors with various degrees of fluency and experience in writing and editing, it would not be surprising if some parts of the texts are not perfect. In the process of preparing the gold standard annotations for the WikiWars corpus, we have made the following observations.

Document ID	Tokens	TIMEX2	$\frac{\text{Tokens}}{\text{TIMEX2}}$
01_WW2	5,593	169	33.1
02_WW1	10,370	264	39.3
03_AmCivWar	3,529	75	47.1
04_AmRevWar	5,695	146	39.0
05_VietnamWar	11,640	243	47.9
06_KoreanWar	5,992	147	40.8
07_IraqWar	8,404	247	34.0
08_FrenchRev	9,631	174	55.4
09_GrecoPersian	7,393	129	57.3
10_PunicWars	3,475	57	61.0
11_ChineseCivWar	3,905	103	37.9
12_IranIraq	4,508	98	46.0
13_RussianCivWar	3,924	103	38.1
14_FirstIndochinaWar	3,085	70	44.1
15_MexicanRev	3,910	77	50.8
16_SpanishCivilWar	1,455	63	23.1
17_AlgerianWar	7,716	130	59.4
18_SovietsInAfghanistan	5,306	110	48.2
19_RussoJap	2,760	62	44.5
20_PolishSoviet	5,137	106	48.5
21_NigerianCivilWar	2,091	29	72.1
22_2ndItaloAbyssinianWar	3,949	69	57.2
Total for the whole corpus	119,468	2,671	44.7
Average per document	5,430	121	–
Standard deviation	2,663	63	–

Table 2: Statistics of the Wikipedia War corpus.

4.1 Broken Narratives

In some articles we have found situations where a sentence does not appear to cohere with those on either side of it. This may be the result of a number of modifications made by different authors, or it may be due to a lack of writing skill on the part of the person who wrote the paragraph in question. Example (2) below provides an example of this phenomenon: the sentence about de Gaulle being elected president contains a temporal expression which progresses the temporal focus in the narrative to 1959, but the later context of the article strongly suggests that the subsequent reference to *October* is in fact October 1958.

(2) ALN commandos committed numerous acts of sabotage in France in *August*$_{[1958]}$, and the FLN mounted a desperate campaign of terror in Algeria to intimidate Muslims into boycotting the referendum. Despite threats of reprisal, however, 80 percent of the Muslim electorate turned out to vote in *September*$_{[1958]}$, and of these 96 percent approved the constitution. In *February 1959*, de Gaulle was elected president of the new Fifth Republic. He visited Constantine in

[9] Note that these numbers are for the articles as a whole, and not just the sections which we extracted (although these are usually the major part of the article). Additionally, these edits include both major changes (e.g. adding a new section), minor changes (e.g. correcting a grammar error or adding a comma), vandalism (deletion of the page content or the on-purpose provision of false information) and restoring the page after an act of vandalism has been detected.

[10] See, for example, the manual of style concerning formatting dates and numbers, located at `http://en.wikipedia.org/wiki/Wikipedia:DATE`.

October$_{[1958]}$ to announce a program to end the war and create an Algeria closely linked to France.

It would appear that the reference to *February 1959* is a later addition to the text which has been made without the surrounding text being appropriately revised to accommodate this change. Clearly such instances of incoherence will cause problems for any process that attempts to track the temporal focus.

4.2 Ambiguous Writing

We have also found cases of a lack of precision in writing, which leads to ambiguous statements. Consider the following example:

(3) The Afghan government, having secured a treaty in *December 1978* that allowed them to call on Soviet forces, repeatedly requested the introduction of troops in Afghanistan in *the spring* and *summer of 1979*. They requested Soviet troops to provide security and to assist in the fight against the mujahideen rebels. On *April 14, 1979*, the Afghan government requested that the USSR send 15 to 20 helicopters with their crews to Afghanistan, and on *June 16*, the Soviet government responded and sent a detachment of tanks, BMPs, and crews to guard the government in Kabul and to secure the Bagram and Shindand airfields. In response to this request, an airborne battalion, commanded by Lieutenant Colonel A. Lomakin, arrived at the Bagram Air Base on *July 7*. [...]

After *a month*, the Afghan requests were no longer for individual crews and subunits, but for regiments and larger units. In *July*, the Afghan government requested that two motorized rifle divisions be sent to Afghanistan. *The following day*, they requested an airborne division in addition to the earlier requests.

Here, in the first paragraph there are four temporal expressions related to the Afghan government asking for troops and equipment. There is also one date related to the Soviets' reply to these requests and sending of tanks, and one date related to the arrival of an airborne battalion. The second paragraph starts with *after a month*; the first possible interpretation is that this is a month after the 7th July mentioned in the previous paragraph; i.e. the month would end on the 6th of August. But the following sentence reveals that this is not the case, as it mentions some requests for larger units that were made in July. Usually a narrative progresses forwards in time, not backwards, so the month must start either on 14th April or 16th June: if the second sentence elaborates the first one, then it is a month from 16th June; if it just mentions one of the requests for larger units, then it is probably a month from 14th April.

It is also unclear whether the second paragraph talks about the same request for airborne forces which was mentioned in the first paragraph: both these events are dated July. The phrase *In response to this request* is in fact placed very oddly, as its preceding sentence does not mention any request, but rather talks about the Soviets' response to requests. This may suggest that what at first looks just like a careless and ambiguous use of the expression *after a month* is in fact a larger problem of lack of coherency in these two paragraphs.

4.3 Use of Deictic Expressions

One of the articles, `07_IraqWar`, contained a number of deictic temporal expressions, indicative of the fact that the events described were happening contemporaneously to the time of writing (as is often the case in news stories); for example:

(4) a. Democrats plan to push legislation *this spring* that would force the Iraqi government to spend its own surplus to rebuild.

b. A protester said that despite the approval of the Interim Security pact, the Iraqi people would break it in a referendum *next year*.

Obviously, after some time these expressions will no longer make sense, since there is no 'at-the-time-of-writing' time stamp associated with these sentences: for the reader of a Wikipedia article, the reference date is the time of reading. In the case of the above example, these sentences were written in April and December 2008, respectively.[11] Arguably, these sentences should be corrected, making the temporal expressions fully-specified (e.g. *in spring of 2009* and *in 2009*), or context-dependent (e.g. *in spring of that year* and *the following year*) if there is a context in the article which supports their correct interpretation. Of course, not only the temporal expressions need to be revised, but also the tense and aspect of the verbs used in the sentences. In the gold standard annotations, however, we provided the values by interpreting these expressions with respect to the document time stamp (i.e. `2010-SP` and `2010`), as the text itself does not provide any evidence that other dates were intended.

[11] Somewhat laborious document archaeology allows this information to be extracted from Wikipedia's archive.

Pos	Count	Token class or lexical form	Pos	Count	Token class or lexical form	Pos	Count	Token class or lexical form
1	4650	NUMBER_DIGIT_2	18	222	today	35	49	AMPM
2	1942	:	19	202	NUMBER_DIGIT_1	36	48	ORDINAL_DIGIT
3	1499	-	20	191	last	37	48	?
4	1329	NUMBER_DIGIT_4	21	171	WEEKDAYNAME_ABBR	38	45	recently
5	828	ARTICLE	22	145	NUMBER_DIGIT_8	39	43	year-old
6	765	TEMPORALUNIT	23	113	ago	40	42	later
7	634	TEMPORALUNIT_PLURAL	24	108	former	41	41	tonight
8	555	PREPOSITION	25	96	time	42	39	christmas
9	528	now	26	79	right	43	36	tomorrow
10	411	t	27	69	new	44	36	current
11	403	WEEKDAYNAME	28	69	future	45	35	couple
12	335	NUMBER_WORD	29	67	gmt	46	34	recent
13	329	MONTHNAME	30	65	next	47	33	earlier
14	242	MONTHNAME_ABBR	31	63	past	48	32	and
15	240	DAYPART	32	61	yesterday	49	31	early
16	233	DEMONSTRATIVE	33	59	few	50	31	DIRECT_FREQ
17	224	,	34	50	every	51	31	’s

Table 3: The most frequent tokens in TEs in the ACE 2005 Training corpus.

Pos	Count	Token class or lexical form	Pos	Count	Token class or lexical form	Pos	Count	Token class or lexical form
1	1181	MONTHNAME	18	59	:	35	13	first
2	1157	NUMBER_DIGIT_4	19	51	end	36	11	future
3	674	NUMBER_DIGIT_2	20	49	-	37	11	earlier
4	490	ARTICLE	21	47	late	38	11	.
5	288	PREPOSITION	22	37	DAYPART	39	11	’s
6	221	NUMBER_DIGIT_1	23	36	later	40	9	previous
7	211	TEMPORALUNIT	24	36	former	41	9	christmas
8	206	TEMPORALUNIT_PLURAL	25	32	next	42	8	last
9	165	,	26	27	same	43	8	AMPM
10	133	NUMBER_WORD	27	25	period	44	7	battle
11	99	SEASON	28	22	t	45	7	DIRECT_FREQ
12	98	NUMBER_DIGIT_3	29	20	mid-	46	6	short
13	82	bc	30	18	war	47	6	several
14	76	now	31	18	few	48	6	season
15	70	time	32	14	following	49	6	recent
16	67	early	33	14	ORDINAL_DIGIT	50	6	past
17	63	DEMONSTRATIVE	34	13	s	51	6	”

Table 4: The most frequent tokens in TEs in the WikiWars corpus.

4.4 Use of Time Zone Information

Consider the following example, which comes from the article `01_WW2`:

(5) On *December 7 (December 8 in Asian time zones), 1941*, Japan attacked British and American holdings with near simultaneous offensives against Southeast Asia and the Central Pacific.

The italicized temporal expression is difficult to detect, and it is not clear how it should be annotated. But it is also imprecise with respect to which time zone is intended: Asia encompasses 10 time zones. Therefore it is impossible to fully interpret the expression. Note also that the expression combines a time zone with a date, rather than with a time. While uncommon, this is not incorrect; but the TIMEX2 guidelines do not explicitly allow for this circumstance.

4.5 Quotes Missing a Time Stamp

Occasionally it happens that an article contains a quoted utterance, but there is no indication of when the utterance was made. For example, in the document `05_VietnamWar` we find the following:

(6) Nixon said in an announcement, “I am *tonight* announcing plans for the withdrawal of an additional 150,000 American troops to be completed during *the*

> *spring of next year*. This will bring a total reduction of 265,500 men in our armed forces in Vietnam below the level that existed when we took office *15 months ago*."

It is impossible to determine what dates are meant by the three temporal expressions present in the announcement. In some cases this information may be provided in citation footnotes, but this is not always the case; when this is absent, such expressions can only be annotated at the level of textual extent and a localised, context-dependent semantics.

5 Comparing WikiWars to the ACE Data

A comparison of WikiWars with the ACE corpora reveals some interesting differences.

5.1 Vocabulary Differences

First, we found differences on the level of the lexical triggers that signal the presence of temporal expressions. Because of space limitations, we provide here only the main findings.

Tables 3 and 4 present the 51 most frequent tokens, including punctuation, in the ACE 2005 Training and WikiWars corpus, respectively. Some tokens are combined into what we call **trigger classes**; for example, all weekday names belong to the class `WEEKDAYNAME`.[12]

We can see that there are many classes that fall into the top 51 positions for both corpora, e.g. the names of temporal units (such as *month* and *year*). But there are also clear differences. Month names are the most frequent class in WikiWars, while they are not so frequent in ACE. Similarly, year seasons ranked very highly in WikiWars, but do not figure in the rankings shown for ACE. On the other hand, weekday names are quite frequent in the ACE corpus, but do not occur in the table for WikiWars. This suggests that these corpora make different use of temporal expressions: in WikiWars we find many references to the more distant past, thus the high use of month names, but ACE documents tend to discuss temporally local issues, so they are more likely to refer to days in the weeks preceding and following the reference date.

Looking at individual tokens, we can see that deictic expressions such as *today*, *tonight*, *yesterday* and *tomorrow* are in the top 51 positions for ACE, but almost never occur in WikiWars: there are only three instances of *today*, two of *tomorrow* and one of *tonight* in the corpus, and all of these appear only in quoted speech. Similarly, *ago* occurred 113 times in ACE, but only twice in WikiWars: once in quoted speech, and once used incorrectly instead of *earlier* in a context-dependent expression. Other tokens which are frequent in ACE but rare in WikiWars are *recent*, *recently*, *current* and *currently*.

5.2 Temporal Discourse Structure

A more interesting property that WikiWars exhibits, and which is noticeably absent from the simpler ACE data, is what we might think of as a discourse mechanism for resetting the temporal focus. This is a feature of complex texts in general, rather than something that is specific to Wikipedia as a source. In these cases, the discourse does not follow a single global timeline from the beginning to the end of the document, but is rather divided into subdiscourses which describe separate chains of events that often have common temporal starting points. This is typical in the description of big, often international, conflicts, where one can distinguish several theaters of the war, i.e. the eastern and western theaters.

In most cases the switch to a different 'part of the story' can be determined not only by analysing the events and their geographic locations, but by recognizing that the first date appearing in the new subdiscourse is generally fully specified. This is, however, not always the case, as shown in the following example extracted from the article `01_WW2`:

(7) In northern Serbia, the Red Army, with limited support from Bulgarian forces, assisted the partisans in a joint liberation of the capital city of Belgrade on *October 20*$_{[1944]}$. *A few days later*, the Soviets launched a massive assault against German-occupied Hungary that lasted until the fall of Budapest in *February 1945*. [...]

By *the start of July*$_{[1944]}$, Commonwealth forces in Southeast Asia had repelled the Japanese sieges in Assam, pushing the Japanese back to the Chindwin River while the Chinese captured Myitkyina. In China, the Japanese were having greater successes, having fi-

[12] The entries in the table correspond to the lexical and punctuation clues that drive detection of temporal expressions: the high rank of colons and dashes comes from their use in document time stamps, which are considered markable by the TIMEX2 guidelines. The *T* token is a separator that often occurs in timestamps, e.g. *2005-01-25T11:08:00*; the question mark appears very often because some of the ACE timestamps are of the form *????-??-??T19:33:00*.

> nally captured Changsha in *mid-June*$_{[1944]}$ and the city of Hengyang by *early August*$_{[1944]}$. Soon after, they [. . .] by *the end of November*$_{[1944]}$ and successfully linking up their forces in China and Indochina by *the middle of December*$_{[1944]}$.

Clearly, quite sophisticated processing is required to handle this phenomenon adequately.

6 Automated Processing of WikiWars

After we developed the WikiWars corpus, we used it to evaluate our temporal expression tagger, DANTE, which had been developed for participation in ACE. Performance at finding temporal expressions in text is traditionally reported, for example by (Mani and Wilson, 2000; Negri and Marseglia, 2005; Teissèdre et al., 2010), in terms of precision, recall and F-measure. These can, however, be calculated in two ways, **lenient** and **strict**, corresponding to two tasks: **detection** (where a single character overlap between the gold standard and system annotation counts as a correct answer) and **recognition** (where an exact overlap is required).

Table 5 shows our tagger's initial performance on the data. While the lenient F-measure for extent recognition was comparable to that obtained for the ACE 2005 Training corpus (0.82 vs 0.78), the recall was much lower: 0.75 vs 0.87. The difference in strict results was even larger, where both precision and recall were lower for WikiWars than for ACE, resulting in an F-measure of 0.38. When evaluating also the `VAL` attribute, the strict F-measure was quite low for both corpora, but significantly lower for WikiWars: 0.17 vs 0.33. This illustrates how illusive it may be to trust the performance of a tagger measured on a single, possibly biased, data set.

In the light of the results of our comparison in Section 5, it is clear that at some of the performance loss here is simply due to domain differences with respect to lexical triggers. So, we extended DANTE's coverage with approximately 20 temporal triggers and modifiers to include the more common vocabulary that appeared in the WikiWars data; we also modified the recognition grammar to reduce the number of spurious matches and extent errors. These changes resulted in the improvements shown in Table 6. The performance on extent recognition improves significantly for both sets of data, but the gap between extent recognition and evaluation of the `VAL` attribute is much larger on WikiWars. This is most likely because the strategy of using the document time stamp for the interpretation of context-dependent expressions does not work at all for WikiWars documents, whereas it works well for ACE documents, in line with our earlier comments in regard to the genres of the documents. This emphasises the need to develop sophisticated methods for temporal focus tracking if we are to extend current time-stamping technologies beyond the relatively simplistic temporal structures found in currently available corpora.

	Lenient			Strict		
Corpus and Task	Prec	Rec	F	Prec	Rec	F
WW - Extent only	0.90	0.75	0.82	0.42	0.35	0.38
WW - Extent + VAL	0.22	0.18	0.20	0.19	0.16	0.17
ACE - Extent only	0.71	0.87	0.78	0.53	0.65	0.58
ACE - Extent +VAL	0.34	0.42	0.37	0.30	0.36	0.33

Table 5: Initial performance of DANTE on WikiWars and the ACE 2005 Training corpus.

	Lenient			Strict		
Corpus and Task	Prec	Rec	F	Prec	Rec	F
WW - Extent only	0.98	0.99	0.99	0.95	0.95	0.95
WW - Extent + VAL	0.59	0.60	0.59	0.58	0.59	0.58
ACE - Extent only	0.88	0.93	0.90	0.75	0.79	0.77
ACE - Extent +VAL	0.63	0.67	0.65	0.57	0.60	0.58

Table 6: Current performance of DANTE on WikiWars and the ACE 2005 Training corpus.

7 Conclusions and Future Work

We have presented a new corpus based on the historical descriptions of 22 wars sourced from English Wikipedia, and we have described in detail the methodology adopted to construct the corpus; the corpus can be easily extended in the same way. We annotated temporal expressions in these documents with TIMEX2 tags, which provide both the textual extents and the semantics of the expressions in the context of whole article.

Following an analysis of the differences between our new corpus and existing data sets, we then presented the results of automatic processing of the corpus. This demonstrates that differences in the vocabulary used for temporal expressions can be fairly straightforwardly incorporated in a tagging tool, but that appropriate processing of temporal structure in complex documents requires more sophisticated techniques than those required to handle existing corpora. The WikiWars Corpus provides data that tests these capabilities.

References

David Ahn, Sisay Fissaha Adafre, and Maarten de Rijke. 2005. Recognizing and Interpreting Temporal Expressions in Open Domain Texts. In *We Will Show Them: Essays in Honour of Dov Gabbay, Vol 1*, pages 31–50, October.

David Ahn, Joris van Rantwijk, and Maarten de Rijke. 2007. A cascaded machine learning approach to interpreting temporal expressions. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2007)*, Rochester, NY, USA, April.

Jennifer Baldwin. 2002. Learning Temporal Annotation of French News. Master's thesis, Dept. of Linguistics, Georgetown University, April.

Branimir Boguraev, Jose Castaño, Rob Gaizauskas, Bob Ingria, Graham Katz, Bob Knippen, Jessica Littman, Inderjeet Mani, James Pustejovsky, Antonio Sanfilippo, Andrew See, Andrea Setzer, Roser Saurí, Amber Stubbs, Beth Sundheim, Svetlana Symonenko, and Marc Verhagen. 2005. TimeML 1.2.1 – A Formal Specification Language for Events and Temporal Expressions, October.

Branimir Boguraev, James Pustejovsky, Rie Ando, and Marc Verhagen. 2007. TimeBank evolution as a community resource for TimeML parsing. *Language Resources and Evaluation*, 41(1):91–115, 02.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the ACL*.

Lisa Ferro, L. Gerber, I. Mani, B. Sundheim, and G. Wilson. 2005. TIDES 2005 Standard for the Annotation of Temporal Expressions. Technical report, MITRE, September.

Kadri Hacioglu, Ying Chen, and Benjamin Douglas. 2005. Automatic time expression labeling for english and chinese text. In Alexander F. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing, 6th International Conference, CICLing'05*, Lecture Notes in Computer Science, pages 548–559, Mexico City, Mexico, February. Springer.

Benjamin Han, Donna Gates, and Lori Levin. 2006. From language to time: A temporal expression anchorer. In *Proceedings of the Thirteenth International Symposium on Temporal Representation and Reasoning (TIME'06)*, pages 196–203. IEEE Computer Society, June.

Inderjeet Mani and George Wilson. 2000. Robust temporal processing of news. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics (ACL '00)*, pages 69–76, Morristown, NJ, USA, October. Association for Computational Linguistics.

Pawel Mazur and Robert Dale. 2007. The DANTE Temporal Expression Tagger. In Zygmunt Vetulani, editor, *Proceedings of the 3rd Language And Technology Conference (LTC)*, Poznan, Poland, October.

Pawel Mazur and Robert Dale. 2008. What's the Date? High Accuracy Interpretation of Weekday Names. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 553–560, Manchester, UK, August. Coling 2008 Organizing Committee.

Matteo Negri and Luca Marseglia. 2005. Recognition and normalization of time expressions: Itc-irst at tern 2004. Technical Report WP3.7, Information Society Technologies, February.

Feng Pan, R. Mulkar, and J. R. Hobbs. 2006. Learning event durations from event descriptions. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 393–400, Sydney, Australia, July. Association for Computational Linguistics.

James Pustejovsky, J. Castaño, R. Ingria, R. Saurí, R. Gaizauskas, A. Setzer, and G. Katz. 2003. TimeML: Robust Specification of Event and Temporal Expressions in Text. In *IWCS-5, Fifth International Workshop on Computational Semantics*, Tilburg, The Netherlands, January.

James Pustejovsky, Kiyong Lee, Harry Bunt, and Laurent Romary. 2010. ISO-TimeML: An International Standard for Semantic Annotation. In Bente Maegaard Joseph Mariani Jan Odjik Stelios Piperidis Mike Rosner Daniel Tapias Nicoletta Calzolari (Conference Chair), Khalid Choukri, editor, *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May. European Language Resources Association (ELRA).

Estela Saquete. 2005. *Temporal Expression Recognition and Resolution applied to Event Ordering*. Ph.D. thesis, Departamento de Lenguages y Sistemas Informaticos, Universidad de Alicante, June.

Frank Schilder. 2004. Extracting meaning from temporal nouns and temporal prepositions. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(1):33–50, March.

Charles Teissèdre, Delphine Battistelli, and Jean-Luc Minel. 2010. Resources for calendar expressions semantic tagging and temporal navigation through texts. In *Proceedings of LREC2010*, May.

PEM: A Paraphrase Evaluation Metric Exploiting Parallel Texts

Chang Liu[1] and **Daniel Dahlmeier**[2] and **Hwee Tou Ng**[1,2]
[1]Department of Computer Science, National University of Singapore
[2]NUS Graduate School for Integrative Sciences and Engineering
{liuchan1,danielhe,nght}@comp.nus.edu.sg

Abstract

We present PEM, the first fully automatic metric to evaluate the quality of paraphrases, and consequently, that of paraphrase generation systems. Our metric is based on three criteria: *adequacy*, *fluency*, and *lexical dissimilarity*. The key component in our metric is a robust and shallow semantic similarity measure based on pivot language N-grams that allows us to approximate adequacy independently of lexical similarity. Human evaluation shows that PEM achieves high correlation with human judgments.

1 Introduction

In recent years, there has been an increasing interest in the task of paraphrase generation (PG) (Barzilay and Lee, 2003; Pang et al., 2003; Quirk et al., 2004; Bannard and Callison-Burch, 2005; Kauchak and Barzilay, 2006; Zhao et al., 2008; Zhao et al., 2009). At the same time, the task has seen applications such as machine translation (MT) (Callison-Burch et al., 2006; Madnani et al., 2007; Madnani et al., 2008), MT evaluation (Kauchak and Barzilay, 2006; Zhou et al., 2006a; Owczarzak et al., 2006), summary evaluation (Zhou et al., 2006b), and question answering (Duboue and Chu-Carroll, 2006).

Despite the research activities, we see two major problems in the field. First, there is currently no consensus on what attributes characterize a good paraphrase. As a result, works on the application of paraphrases tend to build their own PG system in view of the immediate needs instead of using an existing system.

Second, and as a consequence, no automatic evaluation metric exists for paraphrases. Most works in this area resort to ad hoc manual evaluations, such as the percentage of “yes” judgments to the question of “is the meaning preserved”. This type of evaluation is incomprehensive, expensive, and non-comparable between different studies, making progress hard to judge.

In this work we address both problems. We propose a set of three criteria for good paraphrases: *adequacy*, *fluency*, and *lexical dissimilarity*. Considering that paraphrase evaluation is a very subjective task with no rigid definition, we conduct experiments with human judges to show that humans generally have a consistent intuition for good paraphrases, and that the three criteria are good indicators.

Based on these criteria, we construct PEM (Paraphrase Evaluation Metric), a fully automatic evaluation metric for PG systems. PEM takes as input the original sentence R and its paraphrase candidate P, and outputs a single numeric score b estimating the quality of P as a paraphrase of R. PG systems can be compared based on the average scores of their output paraphrases. To the best of our knowledge, this is the first automatic metric that gives an objective and unambiguous ranking of different PG systems, which serves as a benchmark of progress in the field of PG.

The main difficulty of deriving PEM is to measure semantic closeness without relying on lexical level similarity. To this end, we propose *bag of pivot language N-grams* (*BPNG*) as a robust, broad-coverage, and knowledge-lean semantic representation for natural language sentences. Most importantly, BPNG does not depend on lexical or syntactic similarity, allowing us to address the conflicting requirements of paraphrase evaluation. The only linguistic re-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 923–932,
MIT, Massachusetts, USA, 9-11 October 2010.

source required to evaluate BPNG is a parallel text of the target language and an arbitrary other language, known as the pivot language.

We highlight that paraphrase evaluation and paraphrase recognition (Heilman and Smith, 2010; Das and Smith, 2009; Wan et al., 2006; Qiu et al., 2006) are related yet distinct tasks. Consider two sentences S_1 and S_2 that are the same except for the substitution of a single synonym. A paraphrase recognition system should assign them a very high score, but a paraphrase evaluation system would assign a relatively low one. Indeed, the latter is often a better indicator of how useful a PG system potentially is for the applications of PG described earlier.

The rest of the paper is organized as follows. We survey other automatic evaluation metrics in natural language processing (NLP) in Section 2. We define the task of paraphrase evaluation in Section 3 and develop our metric in Section 4. We conduct a human evaluation and analyze the results in Section 5. The correlation of PEM with human judgments is studied in Section 6. Finally, we discuss our findings and future work in Section 7 and conclude in Section 8.

2 Related work

The most well-known automatic evaluation metric in NLP is BLEU (Papineni et al., 2002) for MT, based on N-gram matching precisions. The simplicity of BLEU lends well to MT techniques that directly optimize the evaluation metric.

The weakness of BLEU is that it operates purely at the lexical surface level. Later works attempt to take more syntactic and semantic features into consideration (see (Callison-Burch et al., 2009) for an overview). The whole spectrum of NLP resources has found application in machine translation evaluation, including POS tags, constituent and dependency parses, WordNet (Fellbaum, 1998), semantic roles, textual entailment features, and more. Many of these metrics have been shown to correlate better with human judges than BLEU (Chan and Ng, 2008; Liu et al., 2010). Interestingly, few MT evaluation metrics exploit parallel texts as a source of information, when statistical MT is centered almost entirely around mining parallel texts.

Compared to these MT evaluation metrics, our method focuses on addressing the unique requirement of paraphrase evaluation: that lexical closeness does not necessarily entail goodness, contrary to the basis of MT evaluation.

Inspired by the success of automatic MT evaluation, Lin (2004) and Hovy et al. (2006) propose automatic metrics for summary evaluation. The former is entirely lexical based, whereas the latter also exploits constituent and dependency parses, and semantic features derived from WordNet.

The only prior attempt to devise an automatic evaluation metric for paraphrases that we are aware of is ParaMetric (Callison-Burch et al., 2008), which compares the collection of paraphrases discovered by automatic paraphrasing algorithms against a manual gold standard collected over the same sentences. The recall and precision of several current paraphrase generation systems are evaluated. ParaMetric does not attempt to propose a single metric to correlate well with human judgments. Rather, it consists of a few indirect and partial measures of the quality of PG systems.

3 Task definition

The first step in defining a paraphrase evaluation metric is to define a good paraphrase. Merriam-Webster dictionary gives the following definition: *a restatement of a text, passage, or work giving the meaning in another form*. We identify two key points in this definition: (1) that the meaning is preserved, and (2) that the lexical form is different. To which we add a third, that the paraphrase must be fluent.

The first and last point are similar to MT evaluation, where *adequacy* and *fluency* have been established as the standard criteria. In paraphrase evaluation, we have one more: *lexical dissimilarity*. Although lexical dissimilarity is seemingly the easiest to judge automatically among the three, it poses an interesting challenge to automatic evaluation metrics, as overlap with the reference has been the basis of almost all evaluation metrics. That is, while MT evaluation and paraphrase evaluation are conceptually closely related, the latter actually highlights the deficiencies of the former, namely that in most automatic evaluations, semantic equivalence is under-represented and substituted by lexical and syntactic equivalence.

The task of paraphrase evaluation is then defined as follows: Given an original sentence R and a paraphrase candidate P, output a numeric score b estimating the quality of P as a paraphrase of R by considering adequacy, fluency, and lexical dissimilarity. In this study, we use a scale of 1 to 5 (inclusive) for b, although that can be transformed linearly into any range desired.

We observe here that the overall assessment b is not a linear combination of the three measures. In particular, a high dissimilarity score is meaningless by itself. It could simply be that the paraphrase is unrelated to the source sentence, or is incoherent. However, when accompanied by high adequacy and fluency scores, it differentiates the mediocre paraphrases from the good ones.

4 Paraphrase Evaluation Metric (PEM)

In this section we devise our metric according to the three proposed evaluation criteria, namely adequacy, fluency, and dissimilarity. The main challenge is to measure the adequacy, or semantic similarity, completely independent of any lexical similarity. We address this problem in Sections 4.1 to 4.3. The remaining two criteria are addressed in Section 4.4, and we describe the final combined metric PEM in Section 4.5.

4.1 Phrase-level semantic representation

Without loss of generality, suppose we are to evaluate English paraphrases, and have been supplied many sentence-aligned parallel texts of French and English as an additional resource. We can then align the parallel texts at word level automatically using well-known algorithms such as GIZA++ (Och and Ney, 2003) or the Berkeley aligner (Liang et al., 2006; Haghighi et al., 2009).

To measure adequacy without relying on lexical similarity, we make the key observation that the aligned French texts can act as a proxy of the semantics to a fragment of an English text. If two English phrases are often mapped to the same French phrase, they can be considered similar in meaning. Similar observations have been made by previous researchers (Wu and Zhou, 2003; Bannard and Callison-Burch, 2005; Callison-Burch et al., 2006; Snover et al., 2009). We can treat the distribution of aligned French phrases as a semantic representation of the English phrase. The semantic distance between two English phrases can then be measured by their degree of overlap in this representation.

In this work, we use the widely-used phrase extraction heuristic in (Koehn et al., 2003) to extract phrase pairs from parallel texts into a phrase table[1]. The phrases extracted do not necessarily correspond to the speakers' intuition. Rather, they are units whose boundaries are preserved during translation. However, the distinction does not affect our work.

4.2 Segmenting a sentence into phrases

Having established a way to measure the similarity of two English phrases, we now extend the concept to sentences. Here we discuss how to segment an English sentence (the original or the paraphrase) into phrases.

From the phrase table, we know the frequencies of all the phrases and we approximate the probability of a phrase p by:

$$Pr(p) = \frac{N(p)}{\sum_{p'} N(p')} \tag{1}$$

$N(\cdot)$ is the count of a phrase in the phrase table, and the denominator is a constant for all p. We define the likelihood of segmenting a sentence S into a sequence of phrases $(p_1, p_2, \ldots, p_n)$ by:

$$Pr(p_1, p_2, \ldots, p_n | S) = \frac{1}{Z(S)} \prod_{i=1}^{n} Pr(p_i) \tag{2}$$

where $Z(S)$ is a normalizing constant. The best segmentation of S according to Equation 2 can be calculated efficiently using a dynamic programming algorithm. Note that $Z(S)$ does not need to be calculated, as it is the same for all different segmentations of S. The formula has a strong preference for longer phrases, since every $Pr(p_i)$ has a large denominator.

Many sentences are impossible to segment into known phrases, including all those containing out-of-vocabulary words. We therefore allow any single word w to be considered as a phrase, and if $N(w) = 0$, we use $N(w) = 0.5$ instead.

[1] The same heuristic is used in the popular MT package Moses.

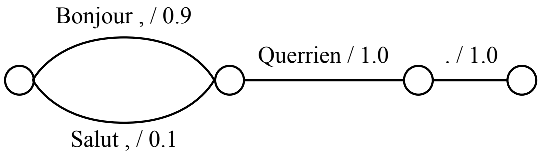


Figure 1: A confusion network in the pivot language

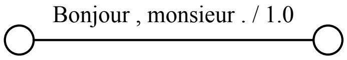


Figure 2: A degenerated confusion network in the pivot language

4.3 Sentence-level semantic representation

Simply merging the phrase-level semantic representations is insufficient to produce a sensible sentence-level semantic representation. For example, assume the English sentence *Morning , sir .* is segmented as a single phrase, because the following phrase pair is found in the phrase table:

En: Morning , sir .

Fr: Bonjour , monsieur .

However, another English sentence *Hello , Querrien .* has an out-of-vocabulary word *Querrien* and consequently the most probable segmentation is found to be "*Hello ,* ||| *Querrien* ||| .":

En: Hello ,

Fr: Bonjour , $(Pr(\text{Bonjour ,}|\text{Hello ,}) = 0.9)$

Fr: Salut , $(Pr(\text{Salut ,}|\text{Hello ,}) = 0.1)$

En: Querrien

Fr: Querrien

En: .

Fr: .

A naive comparison of the bags of French phrases aligned to *Morning , sir .* and *Hello , Querrien .* depicted above would conclude that the two sentences are completely unrelated, as their bags of aligned French phrases are completely disjoint. We tackle this problem by constructing a confusion network representation of the French phrases, as shown in Figures 1 and 2. The confusion network is formed by first joining the different French translations of every English phrase in parallel, and then joining these segments in series.

The confusion network is a compact representation of an exponentially large number of (likely malformed) weighted French sentences. We can easily enumerate the N-grams from the confusion network representation and collect the statistics for this ensemble of French sentences efficiently. In this work, we consider N up to 4. The N-grams for *Hello , Querrien .* are:

1-grams: Bonjour (0.9), Salut (0.1), *comma* (1.0), Querrien (1.0), *period* (1.0).

2-grams: Bonjour *comma* (0.9), Salut *comma* (0.1), *comma* Querrien (1.0), Querrien *period* (1.0).

3-grams: Bonjour *comma* Querrien (0.9), Salut *comma* Querrien (0.1), *comma* Querrien *period* (1.0).

4-grams: Bonjour *comma* Querrien *period* (0.9), Salut *comma* Querrien *period* (0.1).

We call this representation of an English sentence a *bag of pivot language N-grams* (BPNG), where French is the pivot language in our illustrating example. We can extract the BPNG of *Morning , sir .* analogously:

1-grams: Bonjour (1.0), *comma* (1.0), monsieur (1.0), *period* (1.0).

2-grams: Bonjour *comma* (1.0), *comma* monsieur (1.0), monsieur *period* (1.0).

3-grams: Bonjour *comma* monsieur (1.0), *comma* monsieur *period* (1.0).

4-grams: Bonjour *comma* monsieur *period* (1.0).

The BPNG of *Hello , Querrien.* can now be compared sensibly with that of the sentence *Morning , sir .* We use the F$_1$ agreement between the two BPNGs as a measure of the semantic similarity. The F$_1$ agreement is defined as

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The precision and the recall for an original sentence R and a paraphrase P is defined as follows. Let French N-gram $g \in$ BPNG$(R) \cup$BPNG(P), and $W_R(g)$ and $W_P(g)$ be the weights of g in the BPNG of R and P respectively, then

$$\text{Precision} = \frac{\sum_g \min(W_R(g), W_P(g))}{\sum_g W_P(g)}$$

$$\text{Recall} = \frac{\sum_g \min(W_R(g), W_P(g))}{\sum_g W_R(g)}$$

In our example, the numerators for both the precision and the recall are $0.9 + 1 + 1 + 0.9$, for the N-grams Bonjour, *comma*, *period*, and Bonjour *comma* respectively. The denominators for both terms are 10.0. Consequently, F_1 = Precision = Recall = 0.38, and we conclude that the two sentences are 38% similar. We call the resulting metric the *pivot language* F_1. Note that since F_1 is symmetric with respect to the precision and the recall, our metric is unaffected whether we consider *Morning, sir.* as the paraphrase of *Hello, Querrien .* or the other way round.

An actual example from our corpus is:

Reference sihanouk ||| put forth ||| this proposal ||| in ||| a statement ||| made ||| yesterday ||| .

Paraphrase shihanuk ||| put forward ||| this proposal ||| in his ||| yesterday ||| 's statement ||| .

The ||| sign denotes phrase segmentation as described earlier. Our semantic representation successfully recognizes that *put forth* and *put forward* are paraphrases of each other, based on their similar Chinese translation statistics (*ti2 chu1* in Chinese).

4.4 Fluency and dissimilarity

We measure the fluency of a paraphrase by a normalized language model score P_n, defined by

$$P_n = \frac{\log Pr(S)}{\text{length}(S)}$$

where $Pr(S)$ is the sentence probability predicted by a standard 4-gram language model.

We measure dissimilarity between two English sentences using the *target language* F_1, where we collect the bag of all N-grams up to 4-grams from each English (referred to as the target language) sentence. The target language F_1 is then defined as the F_1 agreement of the two bags of N-grams, analogous to the definition of the pivot language F_1. The target language F_1 correlates positively with the similarity of the two sentences, or equivalently, negatively with the dissimilarity of the two sentences.

4.5 The metric

To produce the final PEM metric, we combine the three component automatic metrics, pivot language F_1, normalized language model, and target language F_1, which measure adequacy, fluency, and dissimilarity respectively.

As discussed previously, a linear combination of the three component metrics is insufficient. We turn to support vector machine (SVM) regression with the radial basis function (RBF) kernel. The RBF is a simple and expressive function, commonly used to introduce non-linearity into large margin classifications and regressions.

$$\text{RBF}(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2}$$

We use the implementation in SVM^{light} (Joachims, 1999). The SVM is to be trained on a set of human-judged paraphrase pairs, where the three component automatic metrics are fit to the human overall assessment. After training, the model can then be used to evaluate new paraphrase pairs in a fully automatic fashion.

5 Human evaluation

To validate our definition of paraphrase evaluation and the PEM method, we conduct an experiment to evaluate paraphrase qualities manually, which allows us to judge whether paraphrase evaluation according to our definition is an inherently coherent and well-defined problem. The evaluation also allows us to establish an upper bound for the paraphrase evaluation task, and to validate the contribution of the three proposed criteria to the overall paraphrase score.

5.1 Evaluation setup

We use the Multiple-Translation Chinese Corpus (MTC)[2] as a source of paraphrases. The MTC corpus consists of Chinese news articles (993 sentences in total) and multiple sentence-aligned English translations. We select one human translation as the original text. Two other human translations and two automatic machine translations serve as paraphrases of the original sentences. We refer to the two human translations and the two MT systems as paraphrase systems *human1, human2, machine1,* and *machine2*.

We employ three human judges to manually assess the quality of 300 original sentences paired with each of the four paraphrases. Therefore, each judge assesses 1,200 paraphrase pairs in total. The

[2] LDC Catalog No.: LDC2002T01

judgment for each paraphrase pair consists of four scores, each given on a five-point scale:

- Adequacy (*Is the meaning preserved adequately?*)
- Fluency (*Is the paraphrase fluent English?*)
- Lexical Dissimilarity (*How much has the paraphrase changed the original sentence?*)
- Overall score

The instructions given to the judges for the overall score were as follows.

> *A good paraphrase should convey the same meaning as the original sentence, while being as different as possible on the surface form and being fluent and grammatical English. With respect to this definition, give an overall score from 5 (perfect) to 1 (unacceptable) for this paraphrase.*

The paraphrases are presented to the judges in a random order and without any information as to which paraphrase system produced the paraphrase.

In addition to the four paraphrase systems mentioned above, for each original English sentence, we add three more artificially constructed paraphrases with pre-determined "human" judgment scores: (1) the original sentence itself, with adequacy 5, fluency 5, dissimilarity 1, and overall score 2; (2) a random sentence drawn from the same domain, with adequacy 1, fluency 5, dissimilarity 5, and overall score 1; and (3) a random sentence generated by a unigram language model, with adequacy 1, fluency 1, dissimilarity 5, and overall score 1. These artificial paraphrases serve as controls in our evaluation. Our final data set therefore consists of 2,100 paraphrase pairs with judgments on 4 different criteria.

5.2 Inter-judge correlation

The first step in our evaluation is to investigate the correlation between the human judges. We use Pearson's correlation coefficient, a common measure of the linear dependence between two random variables.

We investigate inter-judge correlation at the sentence and at the system level. At the sentence level, we construct three vectors, each containing the 1,200 sentence level judgments from one judge for the overall score. The pair-wise correlations between these three vectors are then taken. Note that we exclude the three artificial control paraphrase systems from consideration, as that would inflate the correlation. At the system level, we construct three vectors each of size four, containing the average scores given by one judge to each of the four paraphrase systems human1, human2, machine1, and machine2. The correlations are then taken in the same fashion.

	Sentence Level		System Level	
	Judge A	Judge B	Judge A	Judge B
Judge B	0.6406	-	0.9962	-
Judge C	0.6717	0.5993	0.9995	0.9943

Table 1: Inter-judge correlation for overall paraphrase score

	Sentence Level	System Level
Adequacy	0.7635	0.7616
Fluency	0.3736	0.3351
Dissimilarity	-0.3737	-0.3937
Dissimilarity (A,F≥4)	0.8881	0.9956

Table 2: Correlation of paraphrase criteria with overall score

The results are listed in Table 1. The inter-judge correlation is between 0.60 and 0.67 at the sentence level and above 0.99 at the system level. These correlation scores can be considered very high when compared to similar results reported in MT evaluations, e.g., Blatz et al. (2003). The high correlation confirms that our evaluation task is well defined.

Having confirmed that human judgments correlate strongly, we combine the scores of the three judges by taking their arithmetic mean. Together with the three artificial control paraphrase systems, they form the human reference evaluation which we use for the remainder of the experiments.

5.3 Adequacy, fluency, and dissimilarity

In this section, we empirically validate the importance of our three proposed criteria: adequacy, fluency, and lexical dissimilarity. This can be done by measuring the correlation of each criterion with the overall score. The system and sentence level correlations are shown in Table 2.

We can see a positive correlation of adequacy and

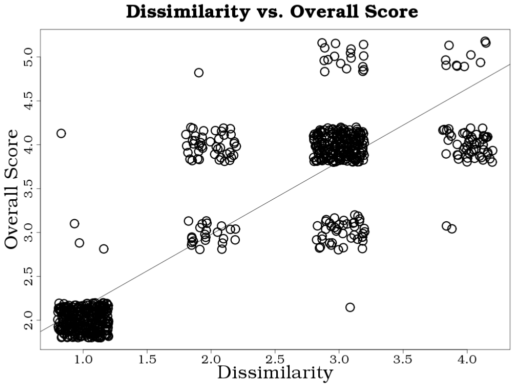


Figure 3: Scatter plot of dissimilarity vs. overall score for paraphrases with high adequacy and fluency.

fluency with the overall score, and the correlation with adequacy is particularly strong. Thus, higher adequacy and to a lesser degree higher fluency indicate higher paraphrase quality to the human judges.

On the other hand, dissimilarity is found to have a negative correlation with the overall score. This can be explained by the fact that the two human translations usually have much higher similarity with the reference translation, and at the same time are scored as better paraphrases. This effect dominates a simple linear fitting of the paraphrase score vs. the dissimilarity, resulting in the counter intuitive negative correlation. We note that a high dissimilarity alone tells us little about the quality of the paraphrase. Rather, we expect dissimilarity to be a differentiator between the mediocre and good paraphrases.

To test this hypothesis, we select the subset of paraphrase pairs that receive adequacy and fluency scores of at least four and again measure the correlation of the dissimilarity and the overall score. The result is tabulated in the last row of Table 2 and shows a strong correlation. Figure 3 shows a scatter plot of the same result[3].

The empirical results presented so far confirm that paraphrase evaluation is a well-defined task permitting consistent subjective judgments, and that adequacy, fluency, and dissimilarity are suitable criteria for paraphrase quality.

6 PEM vs. human evaluation

In the last section, we have shown that the three proposed criteria are good indicators of paraphrase quality. In this section, we investigate how well PEM can predict the overall paraphrase quality from the three automatic metrics (pivot language F_1, normalized language model, and target language F_1), designed to match the three evaluation criteria. We describe the experimental setup in Section 6.1, before we show the results in Section 6.2.

6.1 Experimental setup

We build the phrase table used to evaluate the pivot language F_1 from the FBIS Chinese-English corpus, consisting of about 250,000 Chinese sentences, each with a single English translation. The paraphrases are taken from the MTC corpus in the same way as the human experiment described in Section 5.1. Both FBIS and MTC are in the Chinese newswire domain.

We stem all English words in both data sets with the Porter stemmer (Porter, 1980). We use the maximum entropy segmenter of (Low et al., 2005) to segment the Chinese part of the FBIS corpus. Subsequently, word level Chinese-English alignments are generated using the Berkeley aligner (Liang et al., 2006; Haghighi et al., 2009) with five iterations of training. Phrases are then extracted with the widely-used heuristic in Koehn et al. (2003). We extract phrases of up to four words in length.

Bags of Chinese pivot language N-grams are extracted for all paraphrase pairs as described in Section 4.3. For computational efficiency, we consider only edges of the confusion network with probabilities higher than 0.1, and only N-grams with probabilities higher than 0.01 in the bag of N-grams. We collect N-grams up to length four.

The language model used to judge fluency is trained on the English side of the FBIS parallel text. We use SRILM (Stolcke, 2002) to build a 4-gram model with the default parameters.

The PEM SVM regression is trained on the paraphrase pairs for the first 200 original English sentences and tested on the paraphrase pairs of the remaining 100 original English sentences. Thus, there are 1,400 instances for training and 700 instances for testing. For each instance, we calculate the values

[3] We automatically add jitter (small amounts of noise) for ease of presentation.

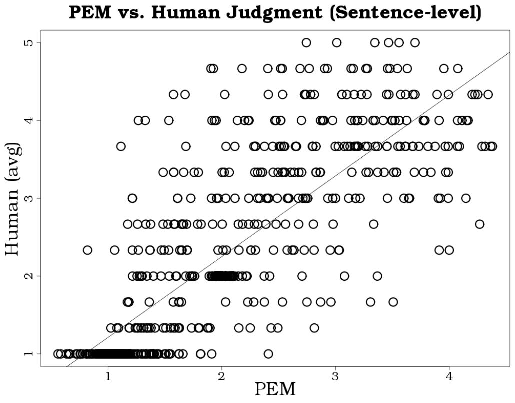


Figure 4: Scatter plot of PEM vs. human judgment (overall score) at the sentence level

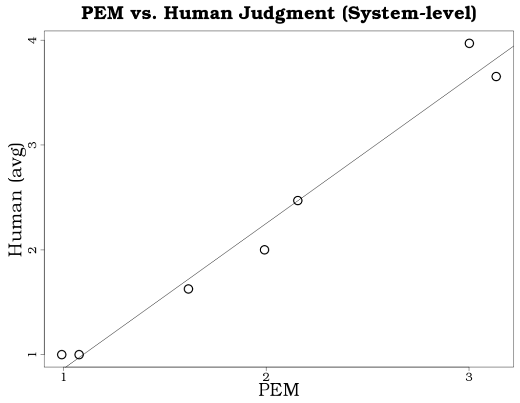


Figure 5: Scatter plot of PEM vs. human judgment (overall score) at the system level

of pivot language F$_1$, normalized language model score, and target language F$_1$. These values serve as the input features to the SVM regression and the target value is the human assessment of the overall score, on a scale of 1 to 5.

6.2 Results

As in the human evaluation, we investigate the correlation of the PEM scores with the human judgments at the sentence and at the system level. Figure 4 shows the sentence level PEM scores plotted against the human overall scores, where each human overall score is the arithmetic mean of the scores given by the three judges. The Pearson correlation between the automatic PEM scores and the human judgments is 0.8073. This is substantially higher than the sentence level correlation of MT metrics like BLEU. For example, the highest sentence level Pearson correlation by any metric in the Metrics-MATR 2008 competition (Przybocki et al., 2009) was 0.6855 by METEOR-v0.6; BLEU achieved a correlation of 0.4513.

	Sentence Level	System Level
PEM vs. Human Avg.	0.8073	0.9867
PEM vs. Judge A	0.5777	0.9757
PEM vs. Judge B	0.5281	0.9892
PEM vs. Judge C	0.5231	0.9718

Table 3: Correlation of PEM with human judgment (overall score)

Figure 5 shows the system level PEM scores plotted against the human scores. The Pearson correlation between PEM scores and the human scores at the system level is 0.9867.

We also calculate the Pearson correlation between PEM and each individual human judge. Here, we exclude the three artificial control paraphrase systems from the data, to make the results comparable to the inter-judge correlation presented in Section 5.2. The correlation is between 0.52 and 0.57 at the sentence level and between 0.97 and 0.98 at the system level. As we would expect, the correlation between PEM and a human judge is not as high as the correlation between two human judges, but PEM still shows a strong and consistent correlation with all three judges. The results are summarized in Table 3.

7 Discussion and future work

The paraphrases that we use in this study are not actual machine generated paraphrases. Instead, the English paraphrases are multiple translations of the same Chinese source sentence. Our seven "paraphrase systems" are two human translators, two machine translation systems, and three artificially created extreme scenarios. The reason for using multiple translations is that we could not find any PG system that can paraphrase a whole input sentence and is publicly available. We intend to obtain and evaluate paraphrases generated from real PG systems and compare their performances in a follow-up study.

Our method models paraphrasing up to the phrase level. Unfortunately, it makes no provisions for syntactic paraphrasing at the sentence level, which is probably a much greater challenge, and the literature offers few successes to draw inspirations from. We hope to be able to partially address this deficiency in future work.

The only external linguistic resource required by PEM is a parallel text of the target language and another arbitrary language. While we only use Chinese-English parallel text in this study, other language pairs need to be explored too. Another alternative is to collect parallel texts against multiple foreign languages, e.g., using Europarl (Koehn, 2005). We leave this for future work.

Our evaluation method does not require human-generated references like in MT evaluation. Therefore, we can easily formulate a paraphrase generator by directly optimizing the PEM metric, although solving it is not trivial:

$$\text{paraphrase}(R) = \arg\max_{P} \text{PEM}(P, R)$$

where R is the original sentence and P is the paraphrase.

Finally, the PEM metric, in particular the semantic representation BPNG, can be useful in many other contexts, such as MT evaluation, summary evaluation, and paraphrase recognition. To facilitate future research, we will package and release PEM under an open source license at `http://nlp.comp.nus.edu.sg/software`.

8 Conclusion

We proposed PEM, a novel automatic metric for paraphrase evaluation based on adequacy, fluency, and lexical dissimilarity. The key component in our metric is a novel technique to measure the semantic similarity of two sentences through their N-gram overlap in an aligned foreign language text. We conducted an extensive human evaluation of paraphrase quality which shows that our proposed metric achieves high correlation with human judgments. To the best of our knowledge, PEM is the first automatic metric for paraphrase evaluation.

Acknowledgments

This research was done for CSIDM Project No. CSIDM-200804 partially funded by a grant from the National Research Foundation (NRF) administered by the Media Development Authority (MDA) of Singapore.

References

C. Bannard and C. Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proc. of ACL*.

R. Barzilay and L. Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proc. of HLT-NAACL*.

J. Blatz, E. Fitzgerald, G. Foster, S. Gandrabur, C. Goutte, A. Kulesza, A. Sanchis, and N. Ueffing. 2003. Confidence estimation for machine translation. Technical report, CLSP Workshop Johns Hopkins University.

C. Callison-Burch, P. Koehn, and M. Osborne. 2006. Improved statistical machine translation using paraphrases. In *Proc. of HLT-NAACL*.

C. Callison-Burch, T. Cohn, and M. Lapata. 2008. ParaMetric: An automatic evaluation metric for paraphrasing. In *Proc. of COLING*.

C. Callison-Burch, P. Koehn, C. Monz, and J. Schroeder. 2009. Findings of the 2009 Workshop on Statistical Machine Translation. In *Proceedings of WMT*.

Y.S. Chan and H.T. Ng. 2008. MAXSIM: A maximum similarity metric for machine translation evaluation. In *Proc. of ACL-08: HLT*.

D. Das and N.A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proc. of ACL-IJCNLP*.

P. Duboue and J. Chu-Carroll. 2006. Answering the question you wish they had asked: The impact of paraphrasing for question answering. In *Proc. of HLT-NAACL Companion Volume: Short Papers*.

C. Fellbaum, editor. 1998. *WordNet: An electronic lexical database*. MIT Press, Cambridge, MA.

A. Haghighi, J. Blitzer, J. DeNero, and D. Klein. 2009. Better word alignments with supervised ITG models. In *Proc. of ACL-IJCNLP*.

M. Heilman and N.A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proc. of NAACL*.

E. Hovy, C.Y. Lin, L. Zhou, and J. Fukumoto. 2006. Automated summarization evaluation with basic elements. In *Proc. of LREC*.

T. Joachims. 1999. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

D. Kauchak and R. Barzilay. 2006. Paraphrasing for automatic evaluation. In *Proc. of HLT-NAACL*.

P. Koehn, F.J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proc. of HLT-NAACL*.

P. Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*, volume 5.

P. Liang, B. Taskar, and D. Klein. 2006. Alignment by agreement. In *Proc. of HLT-NAACL*.

C.Y. Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Proc. of the ACL-04 Workshop on Text Summarization Branches Out*.

C. Liu, D. Dahlmeier, and H.T. Ng. 2010. TESLA: translation evaluation of sentences with linear-programming-based analysis. In *Proc. of WMT*.

J.K. Low, H.T. Ng, and W. Guo. 2005. A maximum entropy approach to Chinese word segmentation. In *Proc. of the 4th SIGHAN Workshop*.

N. Madnani, N.F. Ayan, P. Resnik, and B.J. Dorr. 2007. Using paraphrases for parameter tuning in statistical machine translation. In *Proc. of WMT*.

N. Madnani, P. Resnik, B.J. Dorr, and R. Schwartz. 2008. Are multiple reference translations necessary? Investigating the value of paraphrased reference translations in parameter optimization. In *Proc. of AMTA*.

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

K. Owczarzak, D. Groves, J. Van Genabith, and A. Way. 2006. Contextual bitext-derived paraphrases in automatic MT evaluation. In *Proc. of WMT*.

B. Pang, K. Knight, and D. Marcu. 2003. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proc. of HLT-NAACL*.

K. Papineni, S. Roukos, T. Ward, and W.J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*.

M. Porter. 1980. An algorithm for suffix stripping. *Program*, 40(3).

M. Przybocki, K. Peterson, S. Bronsart, and G Sanders. 2009. Evaluating machine translation with LFG dependencies. *Machine Translation*, 23(2).

L. Qiu, M.Y. Kan, and T.S. Chua. 2006. Paraphrase recognition via dissimilarity significance classification. In *Proc. of EMNLP*.

C. Quirk, C. Brockett, and W. Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proc. of EMNLP*.

M. Snover, N. Madnani, B. Dorr, and R. Schwartz. 2009. Fluency, adequacy, or HTER? Exploring different human judgments with a tunable MT metric. In *Proc. of WMT*.

A. Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proc. of ICSLP*.

S. Wan, M. Dras, R. Dale, and C Paris. 2006. Using dependency-based features to take the 'para-farce' out of paraphrase. In *Proc. of ALTW 2006*.

H. Wu and M. Zhou. 2003. Synonymous collocation extraction using translation information. In *Proc. of ACL*.

S.Q. Zhao, C. Niu, M. Zhou, T. Liu, and S. Li. 2008. Combining multiple resources to improve SMT-based paraphrasing model. In *Proc. of ACL-08: HLT*.

S.Q. Zhao, X. Lan, T. Liu, and S. Li. 2009. Application-driven statistical paraphrase generation. In *Proc. of ACL-IJCNLP*.

L. Zhou, C.Y. Lin, and E. Hovy. 2006a. Re-evaluating machine translation results with paraphrase support. In *Proc. of EMNLP*.

L. Zhou, C.Y. Lin, D.S. Munteanu, and E. Hovy. 2006b. ParaEval: Using paraphrases to evaluate summaries automatically. In *Proc. of HLT-NAACL*.

Assessing Phrase-Based Translation Models with Oracle Decoding

Guillaume Wisniewski and **Alexandre Allauzen** and **François Yvon**
Univ. Paris Sud ; LIMSI—CNRS
91403 ORSAY CEDEX
France
{wisniews,allauzen,yvon}@limsi.fr

Abstract

Extant Statistical Machine Translation (SMT) systems are very complex softwares, which embed multiple layers of heuristics and embark very large numbers of numerical parameters. As a result, it is difficult to analyze output translations and there is a real need for tools that could help developers to better understand the various causes of errors.

In this study, we make a step in that direction and present an attempt to evaluate the quality of the phrase-based translation model. In order to identify those translation errors that stem from deficiencies in the phrase table (PT), we propose to compute the *oracle BLEU-4 score*, that is the best score that a system based on this PT can achieve on a reference corpus. By casting the computation of the oracle BLEU-1 as an Integer Linear Programming (ILP) problem, we show that it is possible to efficiently compute accurate lower-bounds of this score, and report measures performed on several standard benchmarks. Various other applications of these oracle decoding techniques are also reported and discussed.

1 Phrase-Based Machine Translation

1.1 Principle

A Phrase-Based Translation System (PBTS) consists of a ruleset and a scoring function (Lopez, 2009). The ruleset, represented in the phrase table, is a set of phrase[1]pairs $\{(f, e)\}$, each pair expressing that the source phrase f can be rewritten (translated) into a target phrase e. *Translation hypotheses* are generated by iteratively rewriting portions of the source sentence as prescribed by the ruleset, until each source word has been consumed by exactly one rule. The order of target words in an hypothesis is uniquely determined by the order in which the rewrite operation are performed. The *search space* of the translation model corresponds to the set of all possible sequences of rules applications. The scoring function aims to rank all possible translation hypotheses in such a way that the best one has the highest score.

A PBTS is learned from a parallel corpus in two independent steps. In a first step, the corpus is aligned at the word level, by using alignment tools such as Giza++ (Och and Ney, 2003) and some symmetrisation heuristics; phrases are then extracted by other heuristics (Koehn et al., 2003) and assigned numerical weights. In the second step, the parameters of the scoring function are estimated, typically through Minimum Error Rate training (Och, 2003).

Translating a sentence amounts to finding the best scoring translation hypothesis in the search space. Because of the combinatorial nature of this problem, translation has to rely on heuristic search techniques such as greedy hill-climbing (Germann, 2003) or variants of best-first search like multi-stack decoding (Koehn, 2004). Moreover, to reduce the overall complexity of decoding, the search space is typically pruned using simple heuristics. For instance, the state-of-the-art phrase-based decoder Moses (Koehn et al., 2007) considers only a restricted number of translations for each source sequence[2] and enforces a distortion limit[3] over which phrases can be reordered. As a consequence, the best translation hypothesis returned by the decoder is not always the one with the highest score.

1.2 Typology of PBTS Errors

Analyzing the errors of a SMT system is not an easy task, because of the number of models that are combined, the size of these models, and the high complexity of the various decision making processes. For a SMT system, three different kinds of errors can be distinguished (Germann et al., 2004; Auli et al., 2009): *search errors*, *induction errors* and *model errors*. The former corresponds to cases where the hypothesis with the best score is missed by the search procedure, either because of the use of an ap-

[1] Following the usage in statistical machine translation literature, we use “phrase” to denote a subsequence of consecutive words.

[2] the ttl option of Moses, defaulting to 20.

[3] the dl option of Moses, whose default value is 7.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 933–943,
MIT, Massachusetts, USA, 9-11 October 2010.

proximate search method or because of the restrictions of the search space. *Induction errors* correspond to cases where, given the model, the search space does not contain the reference. Finally, *model errors* correspond to cases where the hypothesis with the highest score is not the best translation according to the evaluation metric.

Model errors encompass several types of errors that occur during learning (Bottou and Bousquet, 2008)[4]. *Approximation errors* are errors caused by the use of a restricted and oversimplistic class of functions (here, finite-state transducers to model the generation of hypotheses and a linear scoring function to discriminate them) to model the translation process. *Estimation errors* correspond to the use of sub-optimal values for both the phrase pairs weights and the parameters of the scoring function. The reasons behind these errors are twofold: first, training only considers a finite sample of data; second, it relies on error prone alignments. As a result, some "good" phrases are extracted with a small weight, or, in the limit, are not extracted at all; and conversely that some "poor" phrases are inserted into the phrase table, sometimes with a really optimistic score.

Sorting out and assessing the impact of these various causes of errors is of primary interest for SMT system developers: for lack of such diagnoses, it is difficult to figure out which components of the system require the most urgent attention. Diagnoses are however, given the tight intertwining among the various component of a system, very difficult to obtain: most evaluations are limited to the computation of global scores and usually do not imply any kind of failure analysis.

1.3 Contribution and organization

To systematically assess the impact of the multiple heuristic decisions made during training and decoding, we propose, following (Dreyer et al., 2007; Auli et al., 2009), to work out *oracle scores*, that is to evaluate the best achievable performances of a PBTS. We aim at both studying the expressive power of PBTS and at providing tools for identifying and quantifying causes of failure.

Under standard metrics such as BLEU (Papineni et al., 2002), oracle scores are difficult (if not impossible) to compute, but, by casting the computation of the oracle unigram recall and precision as an Integer Linear Programming (ILP) problem, we show that it is possible to efficiently compute accurate lower-bounds of the oracle BLEU-4 scores and report measurements performed on several standard benchmarks.

The main contributions of this paper are twofold. We first introduce an ILP program able to efficiently find the best hypothesis a PBTS can achieve. This program can be easily extended to test various improvements to phrase-base systems or to evaluate the impact of different parameter settings. Second, we present a number of complementary results illustrating the usage of our oracle decoder for identifying and analyzing PBTS errors. Our experimental results confirm the main conclusions of (Turchi et al., 2008), showing that extant PBTs have the potential to generate hypotheses having very high BLEU-4 score and that their main bottleneck is their scoring function.

The rest of this paper is organized as follows: in Section 2, we introduce and formalize the oracle decoding problem, and present a series of ILP problems of increasing complexity designed so as to deliver accurate lower-bounds of oracle score. This section closes with various extensions allowing to model supplementary constraints, most notably reordering constraints (Section 2.5). Our experiments are reported in Section 3, where we first introduce the training and test corpora, along with a description of our system building pipeline (Section 3.1). We then discuss the baseline oracle BLEU scores (Section 3.2), analyze the non-reachable parts of the reference translations, and comment several complementary results which allow to identify causes of failures. Section 4 discuss our approach and findings with respect to the existing literature on error analysis and oracle decoding. We conclude and discuss further prospects in Section 5.

2 Oracle Decoder

2.1 The Oracle Decoding Problem

Definition To get some insights on the errors of phrase-based systems and better understand their limits, we propose to consider the *oracle decoding problem* defined as follows: given a source sentence, its reference translation[5] and a phrase table, what is the "best" translation hypothesis a system can generate? As usual, the quality of an hypothesis is evaluated by the similarity between the reference and the hypothesis. Note that in the oracle decoding problem, we are only assessing the ability of PBT systems to generate good candidate translations, irrespective of their ability to score them properly.

We believe that studying this problem is interesting for various reasons. First, as described in Section 3.4, comparing the best hypothesis a system could have generated and the hypothesis it actually generates allows us to carry on both quantitative and qualitative failure analysis. The oracle decoding problem can also be used to assess the *expressive power* of phrase-based systems (Auli et al., 2009). Other applications include computing acceptable pseudo-references for discriminative training (Tillmann and Zhang, 2006; Liang et al., 2006; Arun and

[4] We omit here *optimization errors*.

[5] The oracle decoding problem can be extended to the case of multiple references. For the sake of simplicity, we only describe the case of a single reference.

Koehn, 2007) or combining machine translation systems in a multi-source setting (Li and Khudanpur, 2009). We have also used oracle decoding to identify erroneous or difficult to translate references (Section 3.3).

Evaluation Measure To fully define the oracle decoding problem, a measure of the similarity between a translation hypothesis and its reference translation has to be chosen. The most obvious choice is the BLEU-4 score (Papineni et al., 2002) used in most machine translation evaluations.

However, using this metric in the oracle decoding problem raises several issues. First, BLEU-4 is a metric defined at the corpus level and is hard to interpret at the sentence level. More importantly, BLEU-4 is not decomposable[6]: as it relies on 4-grams statistics, the contribution of each phrase pair to the global score depends on the translation of the previous and following phrases and can not be evaluated in isolation. Because of its non-decomposability, maximizing BLEU-4 is hard; in particular, the phrase-level decomposability of the evaluation metric is necessary in our approach.

To circumvent this difficulty, we propose to evaluate the similarity between a translation hypothesis and a reference by the number of their common words. This amounts to evaluating translation quality in terms of unigram precision and recall, which are highly correlated with human judgements (Lavie et al.,). This measure is closely related to the BLEU-1 evaluation metric and the Meteor (Banerjee and Lavie, 2005) metric (when it is evaluated without considering near-matches and the distortion penalty). We also believe that hypotheses that maximize the unigram precision and recall at the sentence level yield corpus level BLEU-4 scores close the maximal achievable. Indeed, in the setting we will introduce in the next section, BLEU-1 and BLEU-4 are highly correlated: as all correct words of the hypothesis will be compelled to be at their correct position, any hypothesis with a high 1-gram precision is also bound to have a high 2-gram precision, etc.

2.2 Formalizing the Oracle Decoding Problem

The oracle decoding problem has already been considered in the case of word-based models, in which all translation units are bound to contain only one word. The problem can then be solved by a *bipartite graph matching* algorithm (Leusch et al., 2008): given a $n \times m$ binary matrix describing possible translation links between source words and target words[7], this algorithm finds the subset of links maximizing the number of words of the reference that have been translated, while ensuring that each word is translated only once.

Generalizing this approach to phrase-based systems amounts to solving the following problem: given a set of possible translation links between potential phrases of the source and of the target, find the subset of links so that the unigram precision and recall are the highest possible. The corresponding oracle hypothesis can then be easily generated by selecting the target phrases that are aligned with one source phrase, disregarding the others. In addition, to mimic the way OOVs are usually handled, we match identical OOV tokens appearing both in the source and target sentences. In this approach, the unigram precision is always one (every word generated in the oracle hypothesis matches exactly one word in the reference). As a consequence, to find the oracle hypothesis, we just have to maximize the recall, that is the number of words appearing both in the hypothesis and in the reference.

Considering phrases instead of isolated words has a major impact on the computational complexity: in this new setting, the optimal segmentations in phrases of both the source and of the target have to be worked out in addition to links selection. Moreover, constraints have to be taken into account so as to enforce a proper segmentation of the source and target sentences. These constraints make it impossible to use the approach of (Leusch et al., 2008) and concur in making the oracle decoding problem for phrase-based models more complex than it is for word-based models: it can be proven, using arguments borrowed from (De Nero and Klein, 2008), that this problem is NP-hard even for the simple unigram precision measure.

2.3 An Integer Program for Oracle Decoding

To solve the combinatorial problem introduced in the previous section, we propose to cast it into an Integer Linear Programming (ILP) problem, for which many generic solvers exist. ILP has already been used in SMT to find the optimal translation for word-based (Germann et al., 2001) and to study the complexity of learning phrase alignments (De Nero and Klein, 2008) models. Following the latter reference, we introduce the following variables: $f_{i,j}$ (resp. $e_{k,l}$) is a binary indicator variable that is true when the phrase contains all spans from between-word position i to j (resp. k to l) of the source (resp. target) sentence. We also introduce a binary variable, denoted $a_{i,j,k,l}$, to describe a possible link between source phrase $f_{i,j}$ and target phrase $e_{k,l}$. These variables are built from the entries of the phrase table according to *selection strategies* introduced in Section 2.4. In the following, index variables are so that:

$$0 \leq i < j \leq n, \text{in the source sentence and}$$
$$0 \leq k < l \leq m, \text{in the target sentence,}$$

[6] Neither at the sentence (Chiang et al., 2008), nor at the phrase level.

[7] The (i, j) entry of the matrix is 1 if the i^{th} word of the source can be translated by the j^{th} word of the reference, 0 otherwise.

where n (resp. m) is the length of the source (resp. target) sentence.

Solving the oracle decoding problem then amounts to optimizing the following objective function:

$$\max_{i,j,k,l} \sum_{i,j,k,l} a_{i,j,k,l} \cdot (l-k)\,, \tag{1}$$

under the constraints:

$$\forall x \in \llbracket 1, m \rrbracket : \sum_{k,l \text{ s.t. } k \leq x \leq l} e_{k,l} \leq 1 \tag{2}$$

$$\forall y \in \llbracket 1, n \rrbracket : \sum_{i,j \text{ s.t. } i \leq y \leq j} f_{i,j} = 1 \tag{3}$$

$$\forall k, l : \sum_{i,j} a_{i,j,k,l} = f_{k,l} \tag{4}$$

$$\forall i, j : \sum_{k,l} a_{i,j,k,l} = e_{i,j} \tag{5}$$

The objective function (1) corresponds to the number of target words that are generated. The first set of constraints (2) ensures that each word in the reference $\mathbf{e}$ appears in no more than one phrase. Maximizing the objective under these constraints amounts to maximizing the unigram recall. The second set of constraints (3) ensures that each word in the source $\mathbf{f}$ is translated exactly once, which guarantees that the search space of the ILP problem is the same as the search space of a phrase-based system. Constraints (4) bind the $f_{k,l}$ and $a_{i,j,k,l}$ variables, ensuring that whenever a link $a_{i,j,k,l}$ is active, the corresponding phrase $f_{k,l}$ is also active. Constraints (5) play a similar role for the reference.

The `Relaxed` Problem Even though it accurately models the search space of a phrase-based decoder, this programs is not really useful as is: due to out-of-vocabulary words or missing entries in the phrase table, the constraint that all source words should be translated yields infeasible problems[8]. We propose to relax this problem and allow some source words to remain untranslated. This is done by replacing constraints (3) by:

$$\forall y \in \llbracket 1, n \rrbracket : \sum_{i,j \text{ s.t. } i \leq y \leq j} f_{i,j} \leq 1$$

To better reflect the behavior of phrase-based decoders, which attempt to translate all source words, we also need to modify the objective function as follows:

$$\sum_{i,j,k,l} a_{i,j,k,l} \cdot (l-k) + \sum_{i,j} f_{i,j} \cdot (j-i) \tag{6}$$

The second term in this new objective ensures that optimal solutions translate as many source words as possible.

The `Relaxed-Distortion` Problem A last caveat with the `Relaxed` optimization program is caused by frequently occurring source tokens, such as function words or punctuation signs, which can often align with more than one target word. For lack of taking distortion information into account in our objective function, all these alignments are deemed equivalent, even if some of them are clearly more satisfactory than others. This situation is illustrated on Figure 1.

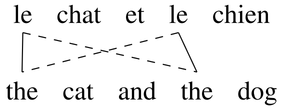


Figure 1: Equivalent alignments between "le" and "the". The dashed lines corresponds to a less interpretable solution.

To overcome this difficulty, we propose a last change to the objective function:

$$\sum_{i,j,k,l} a_{i,j,k,l} \cdot (l-k) + \sum_{i,j} f_{i,j} \cdot (j-i) \\ - \alpha \sum_{i,j,k,l} a_{i,j,k,l} |k-i| \tag{7}$$

Compared to the objective function of the relaxed problem (6), we introduce here a supplementary penalty factor which favors monotonous alignments. For each phrase pair, the higher the difference between source and target positions, the higher this penalty. If α is small enough, this extra term allows us to select, among all the optimal alignments of the `relaxed` problem, the one with the lowest distortion. In our experiments, we set α to $\min\{n, m\}$ to ensure that the penalty factor is always smaller than the reward for aligning two single words.

2.4 Selecting Indicator Variables

In the approach introduced in the previous sections, the oracle decoding problem is solved by selecting, among a set of possible translation links, the ones that yield the solution with the highest unigram recall.

We propose two strategies to build this set of possible translation links. In the first one, denoted *exact match*, an indicator $a_{i,j,k,l}$ is created if there is an entry (f, e) so that f spans from word position i to j in the source and e from word position k to l in the target. In this strategy, the ILP program considers exactly the same ruleset as conventional phrase-based decoders.

We also consider an alternative strategy, which could help us to identify errors made during the phrase extraction process. In this strategy, denoted *inside match*, an indicator $a_{i,j,k,l}$ is created when the following three criteria are met: *i)* f spans from position i to j of the source; *ii)* a *substring* of e, denoted $\bar{e}$, spans from position k to l

[8] An ILP problem is said to be *infeasible* when every possible solution violates at least one constraint.

of the reference; *iii)* $(f, \bar{e})$ is not an entry of the phrase table. The resulting set of indicator variables thus contains, at least, all the variables used in the exact match strategy. In addition, we license here the use of phrases containing words that do not occur in the reference. In fact, using such solutions can yield higher BLEU scores when the reward for additional correct matches exceeds the cost incurred by wrong predictions. These cases are symptoms of situations where the extraction heuristic failed to extract potentially useful subphrases.

2.5 Oracle Decoding with Reordering Constraints

The ILP problem introduced in the previous section can be extended in several ways to describe and test various improvements to phrase-based systems or to evaluate the impact of different parameter settings. This flexibility mainly stems from the possibility offered by our framework to express arbitrary constraints over variables. In this section, we illustrate these possibilities by describing how reordering constraints can easily be considered.

As a first example, the Moses decoder uses a distortion limit to constrain the set of possible reorderings. This constraint "enforces (...) that the last word of a phrase chosen for translation cannot be more than d[9] words from the leftmost untranslated word in the source" (Lopez, 2009) and is expressed as:

$$\forall a_{ijkl}, a_{i'j'k'l'} \text{ s.t. } k > k', \qquad a_{ijkl} \cdot a_{i'j'k'l'} \cdot |j - i' + 1| \leq d,$$

The maximum distortion limit strategy (Lopez, 2009) is also easily expressed and take the following form (assuming this constraint is parameterized by d):

$$\forall l < m - 1, \qquad a_{i,j,k,l} {\cdot} a_{i',j',l+1,l'} \cdot |i' - j - 1| < d$$

Implementing the "local" or MJ-d (Kumar and Byrne, 2005) reordering strategy is also straightforward, and implies using the following constraints:

$$\forall i, k, \left| \sum_{i' \leq i} a_{i',j',k',l'} - \sum_{k' \leq k} a_{i',j',k',l'} \right| \leq d$$

Similarly, It is possible to simulate decoding under the so-called IBM(d) reordering constraints[10] by considering the following constraints:

$$\forall \mu \leq m, \max_{\substack{i,k,l \\ j \leq \mu}} a_{i,j,k,l} \cdot j - \sum_{i,j,k,l} a_{i,j,k,l} \cdot (j - i) \leq d$$

In these constraints, the first factor corresponds to the rightmost translated word of the source and the second one to the number of translated source words. The constraints simply enforce that, at each step of the decoding, there are no more than d source words that were skipped.

Note that the constraints introduced above are not all linear in the problem variables; however they can easily be linearized using standard ILP techniques (Roth and Yih, 2005).

3 Oracle Decoding for Failure Analysis

3.1 Experimental Setting

We propose to use our oracle decoder to study the ability of a PBTS to translate from English to French and from German to English. These two languages pairs present different challenges: English to French translation is considered a relatively easy pair, notwithstanding the difficulties of generating the right inflection marks in French. Translating from German into English is more difficult, notably due to the productivity of inflectional and compounding processes in German, and also to significant differences in word ordering between these languages.

Our experiments are based on the corpora distributed for the WMT'09 constrained tasks (Callison-Burch et al., 2009). All data are tokenized, cleaned and converted to lowercase letters using the tools provided by the organizers. We then used a standard training pipeline to construct the translation model: the bitexts were aligned using Giza++[11], symmetrized using the `grow-diag-final-and` heuristic; the phrase table was extracted and scored using the tools distributed with Moses.[12] Finally, baseline systems were optimized using WMT'08 test set as development using MERT. Note that, for all these steps, we used the default value of the various parameters. The extracted phrase table is then used to find the oracle alignment on the task test set. Recall that oracle decoding do not use the scores estimated by Moses in any way.

In the experiments reported below, two settings are considered. In the first one, denoted NEWSCO, Moses was trained only on a small data set taken from the News Commentary corpus. Using a small sized corpus reduces both training time and decoding time, which allows us to quickly test different configurations of the decoder. In a second setting, denoted EUROPARL, Moses was trained on a larger corpora containing the entirety of the Europarl Corpus, but no in-domain data, to provide results on more realistic conditions. Statistics regarding the different corpora used are reported in Table 1. These statistics are computed on the lowercase cleaned corpora.

[9] This corresponds to the *dl* parameter of Moses

[10] Under IBM(d) constraints, the translation is done, phrase by phrase, from the beginning of the sentence until the end and only one of the first d untranslated phrase can be selected for translation.

[11] http://www.fjoch.com/GIZA++.html

[12] http://statmt.org/moses

	en → fr		de → en	
	NEWSCO	EUROPARL	NEWSCO	EUROPARL
#words	$1,023,401$	$21,616,114$	$1,530,693$	$22,898,644$
#sentences	$51,375$	$1,050,398$	$71,691$	$1,118,399$
#vocabulary	$31,416$	$78,071$	$78,140$	$242,219$
#phrase table	$3,061,701$	$46,003,525$	$4,133,190$	$44,402,367$
% OOV	5.3%	3.1%	8.0%	5.2%

Table 1: Statistics regarding the training corpora: number of words, number of sentences, vocabulary and phrase table size and percentage of test words not appearing in the train set (OOV).

Finding the oracle alignment amounts to solving the ILP problems introduced above. Even though ILP problems are NP-hard in general, there exist several off-the-shelf ILP solvers able to efficiently find an optimal solution or decide that the problem is infeasible. In our experiments, we used the free solver `SCIP` (Achterberg, 2007). An optimal solution was found for all problems we considered. Decoding the $3,027$ sentences of WMT'09 test set takes about 10 minutes (wall time) for the NEWSCO setting, and several hours for the EUROPARL setting[13].

3.2 Oracle BLEU Score

Table 2 reports, for all considered settings, the BLEU-4 scores[14] achieved by our oracle decoder, as well as the number of source words used to generate the oracle hypothesis and the number of target words that are reachable. In these experiments, two objective functions were considered: first, we only consider the objective function corresponding to the relaxed problem defined by Eq. (6); second, we introduced an extra term in the objective to penalize distortion, as described by Eq. (7). Unless explicitly stated otherwise, we always used the exact match strategy.

The main result in 2 is that, for the two language pairs considered, the expressive power of PBTS is not the limiting factor to achieve high translation performance. In fact, for most sentences in the test set, excellent oracle hypotheses, which contain a very high proportion of reference words, are found. This remains true even when the phrase table is extracted from a small corpus. Given that the best BLEU-4 scores achieved during the WMT'09 evaluation are about 28 for the English to French task and 24 for the German to English task ((Callison-Burch et al., 2009), Tables 26 and 25), these results strongly suggest that the main bottleneck of current phrase-based translation systems is their scoring function rather than their expressive power. As we will discuss in Section 4, similar conclusions were drawn by (Auli et al., 2009) and (Turchi et al., 2008).

Several additional comments on these numbers are in order. Despite these very high BLEU scores, in most cases, the reference is only partly translated. In the most favorable case, for the English to French EUROPARL setting, only 26% of the references could be fully generated[15]. These numbers are consistent with the results reported in (Auli et al., 2009). Similarly, only about 31% of the source sentences are completely translated by the oracle decoder, which supports our choice to consider a relaxed version of the ILP problem. Finally, Table 2 also shows that introducing the distortion penalty does not affect the oracle performance of the decoder.

Considering the inside match strategy improves the performance of the oracle decoder: for instance, for the English to French NEWSCO setting, oracle decoder with the inside match strategy achieves a BLEU-4 score of 70.15 (a 2.5 points improvement over the baseline). To achieve this score, 21.45% of the phrases used during decoding were phrases that are not considered by the exact match strategy. Similar results can be observed for other settings, which highlights the significance of one kind of failure of the extraction heuristic: useful "subphrases" of actual phrase pairs are not always extracted.

The numbers in Table 2, no matter how good they may look, should be considered with caution: they only imply that, for most test sentences, all the information necessary to produce a good translation is available in the phrase table. However, the alignment decisions underlying these oracle hypotheses are sometimes hard to justify, and one has to accept that part of these good hypotheses translations are due to a series of lucky alignment errors. This is illustrated on Figure 2, which displays one such lucky oracle alignment based on the misalignment, during training, of the French preposition "des" (*of the*) with the English noun "stock". Such lucky errors are naturally also observed in the outputs of conventional decoders, even though phrase table filtering heuristics probably makes them somewhat more rare.

3.3 Analyzing Non-Reachable Parts of a Reference

Table 3 contains typical examples of sentence pairs that could not be fully generated by our oracle decoder. They

[13] All our experiments are run on a 8 cores computer, each core being a 2.2GHz Intel Processor; the decoder is multi-threaded.

[14] These are computed on lowercase with the default tokenization.

[15] Similar numbers were obtained, albeit much more slowly, with the `--constraint` option of Moses.

	training set	objective function	% source translated	% target generated	4-BLEU
en → fr	NewsCo	Relaxed	86.04%	84.74%	67.65
		Relaxed-Distortion	85.99%	84.77%	67.77
	EuroParl	Relaxed	93.66%	93.06%	85.05
		Relaxed-Distortion	93.65%	93.06%	85.08
de → en	NewsCo	Relaxed	82.57%	82.33%	64.60
		Relaxed-Distortion	82.59%	82.30%	64.65
	EuroParl	Relaxed	90.34%	91.16%	81.77
		Relaxed-Distortion	90.36%	91.12%	81.77

Table 2: Translation score of the ILP oracle decoder for the various settings described in Section 3.1

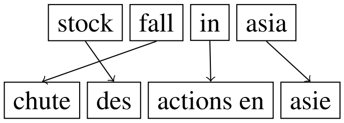


Figure 2: Example of alignment obtained by our oracle decoder

illustrate the three main reasons which cause some parts of the reference to remain *unreachable*:

- phrases are missing from the phrase table, either because they do not occur in the training corpus (OOVs) or because they failed to be extracted. In Table 3, OOV errors are mainly due to past tense forms translated into verbs conjugated in *passé simple* (“rejeta”, “rencontrèrent”, “renoua”) a French literary tense, mostly used in formal writings.
- obvious errors (misspelled words, misinterpretation or mistranslation, ...) in the reference. The reference of the fifth example contains one such error: the state name “Nevada” is translated to “n’évadiez” (literally “have not escaped”), yielding a very poor reference sentence.
- parts of the reference have no translation equivalence in the source. This can be either because references are produced in “context” and some pieces of information are moved across sentence boundaries or because these references are non-literal. The fourth example, which seems to be the translation of a title, falls into this category: the French part contains a reference to the context (“les SA” is referring to the bacteria the text is talking about) which is not in the source text. Non-literal translation are illustrated by the third example, where English “Monday” is translated into French “la veille” *(the day before)*.

While the first kind of errors is inherent to the use of a statistical approach, the last two kinds result from the quality of the data used in the evaluation and directly impact both training and evaluation of automatic translation systems: if they should not distort too much comparisons of MT systems, these errors prevent us from assessing the “global” quality of automatic translation and, if similar errors are found in the train set, they make learning harder as some probability mass is wasted to model them.

To provide a more quantitative analysis, we manually looked at all the non-aligned parts of some WMT’09 references and found that out of 800 references, more than 133 contain either an obvious translation error or can not be achieved by a PBTS[16]. Note that, while identifying these errors could be done in many ways, our oracle decoder makes it far easier.

3.4 Identifying Causes of Failure

By comparing the hypotheses found by the oracle decoder and the ones found by the phrase-based decoder, causes of failure can be easily identified. In this section, we will present several measures that allow us to identify and quantify several causes of failure.

Errors Caused by Search Space Pruning Recall from Section 1.1 that Moses uses several heuristics to prune the search space. In particular, there is a distortion limit and a limit on the number of target phrases considered for one source phrase. In this paragraph, we evaluate the impact of these two heuristics on translation quality.

Table 4 presents the average distortion computed on the oracle hypotheses, as well as the percentage of phrases used that have a distortion strictly greater than 6 (the default distortion limit of Moses). All these numbers are obtained by solving the Relaxed-Distortion problem. Surprisingly enough, the average distortion of oracle hypotheses is quite small, even for the German to English task, and the distortion constraint seems to be violated only in a few cases. It also appears that the distortion of the hypotheses generated in the NewsCo setting is significantly larger than in the EuroParl setting. This can be explained by the extra degrees of freedom in the

[16] Annotation at a finer level is an on-going effort; the annotated corpus is available from http://www.limsi.fr/Individu/wisniews/oracle_decoding.

①	– On Monday the American House of Representatives **rejected** the plan to support the financial system, into which up to 700 billion dollars (nearly 12 **billion Czech crowns**) was to be invested. – Lundi, la chambre des représentants américaine *rejeta* le projet de soutient du système financier, auquel elle aurait dû consacrer jusqu'à 700 milliards de dollars (près de 12 *bilions* de *kč*).
②	– Representatives of the legislators met with American Finance Minister Henry Paulson Saturday night in order to give the government fund a final form. – Dans la nuit de samedi à **dimanche**, des représentants des législateurs *rencontrèrent* le ministre des finances américain Henry Paulson, afin de donner au fond du gouvernement une forme finale.
③	– The Prague Stock Market immediately continued its fall from **Monday** at the beginning of Tuesday's trading , when it dropped by nearly six percent. – Mardi, dès le début des échanges, la bourse de prague *renoua* avec sa chute de la **veille**, lorsqu'elle **perdait** presque six pour **cent**.
④	– **Antibiotic Resistance** – **Les SA résistent aux antibiotiques.**
⑤	– According to **Nevada** Democratic senator Harry Reid, that is how that legislators are trying to have Congress to reach a definitive agreement as early as on Sunday. – D'après le sénateur dèmocrate n'*évadiez* Harry Reid, les législateurs font de sorte que le Congrès **aboutisse** à un accord définitif dès dimanche.

Table 3: Output examples of our oracle decoder on the English to French task. Words in bold are non-aligned words and words in italic are non-aligned out-of-vocabulary words. For clarity the examples have been detokenized and recased.

	training set	avg. distortion	%phrases with a dist. > 6
en → fr	NEWSCO	4.57	22.02%
	EUROPARL	3.21	13.32%
de → en	NEWSCO	5.16	25.37%
	EUROPARL	3.81	17.21%

Table 4: Average distortion and percentage of phrases with a distortion greater that Moses default distortion limit.

alignment decisions enabled by the use of larger training corpora and phrase table.

To evaluate the impact of the second heuristic, we computed the number of phrases discarded by Moses (because of the default *ttl* limit) but used in the oracle hypotheses. In the English to French NEWSCO setting, they account for 34.11% of the total number of phrases used in the oracle hypotheses. When the oracle decoder is constrained to use the same phrase table as Moses, its BLEU-4 score drops to 42.78. This shows that filtering the phrase table prior to decoding discards many useful phrase pairs and is seriously limiting the best achievable performance, a conclusion shared with (Auli et al., 2009).

Search Errors Search errors can be identified by comparing the score of the best hypothesis found by Moses and the score of the oracle hypothesis. If the score of the oracle hypothesis is higher, then there has been a search error; on the contrary, there has been an estimation error when the score of the oracle hypothesis is lower than the score of the best hypothesis found by Moses.

Based on the comparison of the score of Moses hypotheses and of oracle hypotheses for the English to French NEWSCO setting, our preliminary conclusion is that the number of search errors is quite limited: only about 5% of the hypotheses of our oracle decoder are actually getting a better score than Moses solutions. Again, this shows that the scoring function (model error) is one of the main bottleneck of current PBTS. Comparing these hypotheses is nonetheless quite revealing: while Moses mostly selects phrase pairs with high translation scores and generates monotonous alignments, our ILP decoder uses larger reorderings and less probable phrases to achieve better solutions: on average, the reordering score of oracle solutions is -5.74, compared to -76.78 for Moses outputs. Given the weight assigned through MERT training to the distortion score, no wonder that these hypotheses are severely penalized.

The Impact of Phrase Length The observed outputs do not only depend on decisions made during the search, but also on decisions made during training. One such decision is the specification of maximal length for the source and target phrases. In our framework, evaluating the impact of this decision is simple: it suffices to change the definition of indicator variables so as to consider only alignments between phrases of a given length.

In the English-French NEWSCO setting, the most restrictive choice, when only alignments between single words are authorized, yields an oracle BLEU-4 of 48.68; however, authorizing phrases up to length 2 allows to achieve an oracle value of 66.57, very close to the score achieved when considering all extracted phrases (67.77).

This is corroborated with a further analysis of our oracle alignments, which use phrases whose average source length is 1.21 words (respectively 1.31 for target words). If many studies have already acknowledged the predominance of “small” phrases in actual translations, our oracle scores suggest that, for this language pair, increasing the phrase length limit beyond 2 or 3 might be a waste of computational resources.

4 Related Work

To the best of our knowledge, there are only a few works that try to study the expressive power of phrase-based machine translation systems or to provide tools for analyzing potential causes of failure.

The approach described in (Auli et al., 2009) is very similar to ours: in this study, the authors propose to find and analyze the limits of machine translation systems by studying the *reference reachability*. A reference is reachable for a given system if it can be exactly generated by this system. Reference reachability is assessed using Moses in forced decoding mode: during search, all hypotheses that deviate from the reference are simply discarded. Even though the main goal of this study was to compare the search space of phrase-based and hierarchical systems, it also provides some insights on the impact of various search parameters in Moses, delivering conclusions that are consistent with our main results. As described in Section 1.2, these authors also propose a typology of the errors of a statistical translation systems, but do not attempt to provide methods for identifying them.

The authors of (Turchi et al., 2008) study the learning capabilities of Moses by extensively analyzing learning curves representing the translation performances as a function of the number of examples, and by corrupting the model parameters. Even though their focus is more on assessing the scoring function, they reach conclusions similar to ours: the current bottleneck of translation performances is not the representation power of the PBTS but rather in their scoring functions.

Oracle decoding is useful to compute reachable pseudo-references in the context of discriminative training. This is the main motivation of (Tillmann and Zhang, 2006), where the authors compute high BLEU hypotheses by running a conventional decoder so as to maximize a per-sentence approximation of BLEU-4, under a simple (local) reordering model.

Oracle decoding has also been used to assess the limitations induced by various reordering constraints in (Dreyer et al., 2007). To this end, the authors propose to use a beam-search based oracle decoder, which computes lower bounds of the best achievable BLEU-4 using dynamic programming techniques over finite-state (for so-called local and IBM constraints) or hierarchically structured (for ITG constraints) sets of hypotheses. Even though the numbers reported in this study are not directly comparable with ours[17], it seems that our decoder is not only conceptually much simpler, but also achieves much more optimistic lower-bounds of the oracle BLEU score. The approach described in (Li and Khudanpur, 2009) employs a similar technique, which is to guide a heuristic search in an hypergraph representing possible translation hypotheses with n-gram counts matches, which amounts to decoding with a n-gram model trained on the sole reference translation. Additional tricks are presented in this article to speed-up decoding.

Computing oracle BLEU scores is also the subject of (Zens and Ney, 2005; Leusch et al., 2008), yet with a different emphasis. These studies are concerned with finding the best hypotheses in a word graph or in a consensus network, a problem that has various implications for multi-pass decoding and/or system combination techniques. The former reference describes an exponential approximate algorithm, while the latter proves the NP-completeness of this problem and discuss various heuristic approaches. Our problem is somewhat more complex and using their techniques would require us to built word graphs containing all the translations induced by arbitrary segmentations and permutations of the source sentence.

5 Conclusions

In this paper, we have presented a methodology for analyzing the errors of PBTS, based on the computation of an approximation of the BLEU-4 oracle score. We have shown that this approximation could be computed fairly accurately and efficiently using Integer Linear Programming techniques. Our main result is a confirmation of the fact that extant PBTS systems are expressive enough to achieve very high translation performance with respect to conventional quality measurements. The main efforts should therefore strive to improve on the way phrases and hypotheses are scored during training. This gives further support to attempts aimed at designing context-dependent scoring functions as in (Stroppa et al., 2007; Gimpel and Smith, 2008), or at attempts to perform discriminative training of feature-rich models. (Bangalore et al., 2007).

We have shown that the examination of difficult-to-translate sentences was an effective way to detect errors or inconsistencies in the reference translations, making our approach a potential aid for controlling the quality or assessing the difficulty of test data. Our experiments have also highlighted the impact of various parameters.

Various extensions of the baseline ILP program have been suggested and/or evaluated. In particular, the ILP formalism lends itself well to expressing various constraints that are typically used in conventional PBTS. In

[17] The best BLEU-4 oracle they achieve on Europarl German to English is approximately 48; but they considered a smaller version of the training corpus and the WMT’06 test set.

our future work, we aim at using this ILP framework to systematically assess various search configurations. We plan to explore how replacing non-reachable references with high-score pseudo-references can improve discriminative training of PBTS. We are also concerned by determining how tight is our approximation of the BLEU-4 score is: to this end, we intend to compute the best BLEU-4 score within the n-best solutions of the oracle decoding problem.

Acknowledgments

Warm thanks to Houda Bouamor for helping us with the annotation tool. This work has been partly financed by OSEO, the French State Agency for Innovation, under the Quaero program.

References

Tobias Achterberg. 2007. *Constraint Integer Programming*. Ph.D. thesis, Technische Universität Berlin. http://opus.kobv.de/tuberlin/volltexte/2007/1611/.

Abhishek Arun and Philipp Koehn. 2007. Online learning methods for discriminative training of phrase based statistical machine translation. In *Proc. of MT Summit XI*, Copenhagen, Denmark.

Michael Auli, Adam Lopez, Hieu Hoang, and Philipp Koehn. 2009. A systematic analysis of translation model search spaces. In *Proc. of WMT*, pages 224–232, Athens, Greece.

Satanjeev Banerjee and Alon 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, Ann Arbor, Michigan.

Srinivas Bangalore, Patrick Haffner, and Stephan Kanthak. 2007. Statistical machine translation through global lexical selection and sentence reconstruction. In *Proc. of ACL*, pages 152–159, Prague, Czech Republic.

Léon Bottou and Olivier Bousquet. 2008. The tradeoffs of large scale learning. In *Proc. of NIPS*, pages 161–168, Vancouver, B.C., Canada.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. 2009. Findings of the 2009 Workshop on Statistical Machine Translation. In *Proc. of WMT*, pages 1–28, Athens, Greece.

David Chiang, Steve DeNeefe, Yee Seng Chan, and Hwee Tou Ng. 2008. Decomposability of translation metrics for improved evaluation and efficient algorithms. In *Proc. of ECML*, pages 610–619, Honolulu, Hawaii.

John De Nero and Dan Klein. 2008. The complexity of phrase alignment problems. In *Proc. of ACL: HLT, Short Papers*, pages 25–28, Columbus, Ohio.

Markus Dreyer, Keith B. Hall, and Sanjeev P. Khudanpur. 2007. Comparing reordering constraints for smt using efficient bleu oracle computation. In *NAACL-HLT/AMTA Workshop on Syntax and Structure in Statistical Translation*, pages 103–110, Rochester, New York.

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. 2001. Fast decoding and optimal decoding for machine translation. In *Proc. of ACL*, pages 228–235, Toulouse, France.

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

Ulrich Germann. 2003. Greedy decoding for statistical machine translation in almost linear time. In *Proc. of NAACL*, pages 1–8, Edmonton, Canada.

Kevin Gimpel and Noah A. Smith. 2008. Rich source-side context for statistical machine translation. In *Proc. of WMT*, pages 9–17, Columbus, Ohio.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. of NAACL*, pages 48–54, Edmonton, Canada.

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 *Proc. of ACL, demonstration session*.

Philipp Koehn. 2004. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *Proc. of AMTA*, pages 115–124, Washington DC.

Shankar Kumar and William Byrne. 2005. Local phrase reordering models for statistical machine translation. In *Proc. of HLT*, pages 161–168, Vancouver, Canada.

Alon Lavie, Kenji Sagae, and Shyamsundar Jayaraman. The significance of recall in automatic metrics for MT evaluation. In *In Proc. of AMTA*, pages 134–143, Washington DC.

Gregor Leusch, Evgeny Matusov, and Hermann Ney. 2008. Complexity of finding the BLEU-optimal hypothesis in a confusion network. In *Proc. of EMNLP*, pages 839–847, Honolulu, Hawaii.

Zhifei Li and Sanjeev Khudanpur. 2009. Efficient extraction of oracle-best translations from hypergraphs. In *Proc. of NAACL*, pages 9–12, Boulder, Colorado.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proc. of ACL*, pages 761–768, Sydney, Australia.

Adam Lopez. 2009. Translation as weighted deduction. In *Proc. of EACL*, pages 532–540, Athens, Greece.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Comput. Linguist.*, 29(1):19–51.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of 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. Technical report, Philadelphia, Pennsylvania.

D. Roth and W. Yih. 2005. Integer linear programming inference for conditional random fields. In *Proc. of ICML*, pages 737–744, Bonn, Germany.

Nicolas Stroppa, Antal van den Bosch, and Andy Way. 2007. Exploiting source similarity for smt using context-informed features. In Andy Way and Barbara Gawronska, editors, *Proc. of TMI*, pages 231–240, Skövde, Sweden.

Christoph Tillmann and Tong Zhang. 2006. A discriminative global training algorithm for statistical mt. In *Proc. of ACL*, pages 721–728, Sydney, Australia.

Marco Turchi, Tijl De Bie, and Nello Cristianini. 2008. Learning performance of a machine translation system: a statistical and computational analysis. In *Proc. of WMT*, pages 35–43, Columbus, Ohio.

Richard Zens and Hermann Ney. 2005. Word graphs for statistical machine translation. In *Proc. of the ACL Workshop on Building and Using Parallel Texts*, pages 191–198, Ann Arbor, Michigan.

Automatic Evaluation of Translation Quality for Distant Language Pairs

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, Hajime Tsukada
NTT Communication Science Laboratories, NTT Corporation
2-4 Hikaridai, Seikacho, Sorakugun, Kyoto, 619-0237, Japan
{isozaki,hirao,kevinduh,sudoh,tsukada}@cslab.kecl.ntt.co.jp

Abstract

Automatic evaluation of Machine Translation (MT) quality is essential to developing high-quality MT systems. Various evaluation metrics have been proposed, and BLEU is now used as the de facto standard metric. However, when we consider translation between distant language pairs such as Japanese and English, most popular metrics (e.g., BLEU, NIST, PER, and TER) do not work well. It is well known that Japanese and English have completely different word orders, and special care must be paid to word order in translation. Otherwise, translations with wrong word order often lead to misunderstanding and incomprehensibility. For instance, SMT-based Japanese-to-English translators tend to translate 'A because B' as 'B because A.' Thus, word order is the most important problem for distant language translation. However, conventional evaluation metrics do not significantly penalize such word order mistakes. Therefore, locally optimizing these metrics leads to inadequate translations. In this paper, we propose an automatic evaluation metric based on rank correlation coefficients modified with precision. Our meta-evaluation of the NTCIR-7 PATMT JE task data shows that this metric outperforms conventional metrics.

1 Introduction

Automatic evaluation of machine translation (MT) quality is essential to developing high-quality machine translation systems because human evaluation is time consuming, expensive, and irreproducible. If we have a perfect automatic evaluation metric, we can tune our translation system for the metric.

BLEU (Papineni et al., 2002b; Papineni et al., 2002a) showed high correlation with human judgments and is still used as the de facto standard automatic evaluation metric. However, Callison-Burch et al. (2006) argued that the MT community is overly reliant on BLEU by showing examples of poor performance. For Japanese-to-English (JE) translation, Echizen-ya et al. (2009) showed that the popular BLEU and NIST do not work well by using the system outputs of the NTCIR-7 PATMT (patent translation) JE task (Fujii et al., 2008). On the other hand, ROUGE-L (Lin and Hovy, 2003), Word Error Rate (WER), and IMPACT (Echizen-ya and Araki, 2007) worked better.

In these studies, Pearson's correlation coefficient and Spearman's rank correlation ρ with human evaluation scores are used to measure how closely an automatic evaluation method correlates with human evaluation. This evaluation of automatic evaluation methods is called **meta-evaluation**. In human evaluation, people judge the adequacy and the fluency of each translation.

Denoual and Lepage (2005) pointed out that BLEU assumes word boundaries, which is ambiguous in Japanese and Chinese. Here, we assume the word boundaries given by ChaSen, one of the standard morphological analyzers (http://chasen-legacy.sourceforge.jp/) following Fujii et al. (2008)

In JE translation, most Statistical Machine Translation (SMT) systems translate the Japanese sentence

(J0) *kare wa sono hon wo yonda node sekaishi ni kyoumi ga atta*

which means

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 944–952,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

(R0) he was interested in world history because he read the book

into an English sentence such as

(H0) he read the book because he was interested in world history

in which the cause and the effect are swapped. Why does this happen? The former half of (J0) means "He read the book," and the latter half means "(he) was interested in world history." The middle word "node" between them corresponds to "because." Therefore, SMT systems output sentences like (H0). On the other hand, Rule-based Machine Translation (RBMT) systems correctly give (R0).

In order to find (R0), SMT systems have to search a very large space because we cannot restrict its search space with a small distortion limit. Most SMT systems thus fail to find (R0).

Consequently, the global word order is essential for translation between distant language pairs, and wrong word order can easily lead to misunderstanding or incomprehensibility. Perhaps, some readers do not understand why we emphasize word order from this example alone. A few more examples will clarify what happens when SMT is applied to Japanese-to-English translation. Even the most famous SMT service available on the web failed to translate the following very simple sentence at the time of writing this paper.

Japanese: *meari wa jon wo koroshita.*
Reference: Mary killed John.
SMT output: John killed Mary.

Since it cannot translate such a simple sentence, it obviously cannot translate more complex sentences correctly.

Japanese: *bobu ga katta hon wo jon wa yonda.*
Reference: John read a book that Bob bought.
SMT output: Bob read the book John bought.

Another example is:

Japanese: *bobu wa meari ni yubiwa wo kau tameni, jon no mise ni itta.*
Reference: Bob went to John's store to buy a ring for Mary.
SMT output: Bob Mary to buy the ring, John went to the store.

In this way, this SMT service usually gives incomprehensible or misleading translations, and thus people prefer RBMT services. Other SMT systems also tend to make similar word order mistakes, and special care should be paid to the translation between distant language pairs such as Japanese and English.

Even Japanese people cannot solve this word order problem easily: It is well known that Japanese people are not good at speaking English.

From this point of view, conventional automatic evaluation metrics of translation quality disregard word order mistakes too much. Single-reference BLEU is defined by a geometrical mean of n-gram precisions p_n and is modified by Brevity Penalty (BP) $\min(1, \exp(1 - r/h))$, where r is the length of the reference and h is the length of the hypothesis.

$$\text{BLEU} = \text{BP} \times (p_1 p_2 p_3 p_4)^{1/4}.$$

Its range is [0, 1]. The BLEU score of (H0) with reference (R0) is $1.0 \times (11/11 \times 9/10 \times 6/9 \times 4/8)^{1/4} = 0.740$. Therefore, BLEU gives a very good score to this inadequate translation because it checks only n-grams and does not regard global word order.

Since (R0) and (H0) look similar in terms of fluency, **adequacy** is more important than fluency in the translation between distant language pairs.

Similarly, other popular scores such as NIST, PER, and TER (Snover et al., 2006) also give relatively good scores to this translation. NIST also considers only local word orders (n-grams). PER (Position-Independent Word Error Rate) was designed to disregard word order completely. TER (Snover et al., 2006) was designed to allow phrase movements without large penalties. Therefore, these standard metrics are not optimal for evaluating translation between distant language pairs.

In this paper, we propose an alternative automatic evaluation metric appropriate for distant language pairs. Our method is based on **rank correlation coefficients**. We use them to compare the word ranks in the reference with those in the hypothesis.

There are two popular rank correlation coefficients: Spearman's ρ and Kendall's τ (Kendall, 1975). In Isozaki et al. (2010), we used Kendall's τ to measure the effectiveness of our **Head Finalization** rule as a preprocessor for English-to-Japanese translation, but we measured the quality of translation by using conventional metrics.

It is not clear how well τ works as an automatic evaluation metric of translation quality. Moreover, Spearman's ρ might work better than Kendall's τ. As we discuss later, τ considers only the direction of the rank change, whereas ρ considers the distance of the change.

The first objective of this paper is to examine which is the better metric for distant language pairs. The second objective is to find improvements of these rank correlation-metrics.

Spearman's ρ is based on Pearson's correlation coefficients. Suppose we have two lists of numbers

$$\begin{aligned} \boldsymbol{x} &= [0.1, 0.4, 0.2, 0.6], \\ \boldsymbol{y} &= [0.9, 0.6, 0.2, 0.7]. \end{aligned}$$

To obtain Pearson's coefficients between $\boldsymbol{x}$ and $\boldsymbol{y}$, we use the raw values in these lists. If we substitute their ranks for their raw values, we get

$$\boldsymbol{x'} = [1, 3, 2, 4] \text{ and } \boldsymbol{y'} = [4, 2, 1, 3].$$

Then, Spearman's ρ between $\boldsymbol{x}$ and $\boldsymbol{y}$ is given by Pearson's coefficients between $\boldsymbol{x'}$ and $\boldsymbol{y'}$. This ρ can be rewritten as follows when there is no tie:

$$\rho = 1 - \frac{\sum_i d_i^2}{{}_{n+1}C_3}.$$

Here, d_i indicates the difference in the ranks of the i-th element. Rank distances are **squared** in this formula. Because of this square, we expect that ρ decreases drastically when there is an element that significantly changes in rank. But we are also afraid that ρ may be too severe for alternative good translations.

Since Pearson's correlation metric assumes linearity, nonlinear monotonic functions can change its score. On the other hand, Spearman's ρ and Kendall's τ uses ranks instead of raw evaluation scores, and simple application of monotonic functions cannot change them (use of other operations such as averaging sentence scores can change them).

2 Methodology

2.1 Word alignment for rank correlations

We have to determine word ranks to obtain rank correlation coefficients. Suppose we have:

(R1) `John hit Bob yesterday`
(H1) `Bob hit John yesterday`

The 1st word "John" in R1 becomes the **3**rd word in H1. The 2nd word "hit" in R1 becomes the **2**nd word in H1. The 3rd word "Bob" in R1 becomes the **1**st word in H1. The 4th word "yesterday" in R1 becomes the **4**th word in H1. Thus, we get H1's word order list [3, 2, 1, 4]. The number of all pairs of integers in this list is ${}_4C_2 = 6$. It has three increasing pairs: (3,4), (2,4), and (1,4). Since Kendall's τ is given by:

$$\tau = 2 \times \frac{\text{the number of increasing pairs}}{\text{the number of all pairs}} - 1,$$

H1's τ is $2 \times 3/6 - 1 = 0.0$.

In this case, we can obtain Spearman's ρ as follows: "John" moved by $d_1 = 2$ words, "hit" moved by $d_2 = 0$ words, "Bob" moved by $d_3 = 2$ words, and "yesterday" moved by $d_4 = 0$ words. Therefore, H1's ρ is $1 - (2^2 + 0^2 + 2^2 + 0^2)/{}_5C_3 = 0.2$.

Thus, τ considers only the direction of the movement, whereas ρ considers the distance of the movement. Both ρ and τ have the same range $[-1, 1]$. The main objective of this paper is to clarify which rank correlation is closer to human evaluation scores.

We have to consider the limitation of the rank correlation metrics. They are defined only when there is **one-to-one correspondence**. However, a reference sentence and a hypothesis sentence may have different numbers of words. They may have two or more occurrences of the same word in one sentence. Sometimes, a word in the reference does not appear in the hypothesis, or a word in the hypothesis does not appear in the reference. Therefore, we cannot calculate τ and ρ following the above definitions in general.

Here, we determine the correspondence of words between hypotheses and references as follows. First, we find one-to-one corresponding words. That is, we find words that appear in both sentences and only once in each sentence. Suppose we have:

(R2) `the boy read the book`
(H2) `the book was read by the boy`

By removing non-aligned words by one-to-one correspondence, we get:

(R3) `boy read book`
(H3) `book read boy`

Thus, we lost "`the`." We relax this one-to-one correspondence constraint by using one-to-one corresponding **bigrams**. (R2) and (H2) share "`the boy`" and "`the book`," and we can align these instances of "the" correctly.

(R4) the$_1$ boy$_2$ read$_3$ the$_4$ book$_5$
(H4) the$_4$ book$_5$ read$_3$ the$_1$ boy$_2$

Now, we have five aligned words, and H4's word order is represented by [4, 5, 3, 1, 2].

In returning to H0 and R0, we find that each of these sentences has eleven words. Almost all words are aligned by one-to-one correspondence but "he" is not aligned because it appears twice in each sentence. By considering one-to-one corresponding bigrams ("he was" and "he read"), "he" is aligned as follows.

(R5) he$_1$ was$_2$ interested$_3$ in$_4$ world$_5$ history$_6$ because$_7$ he$_8$ read$_9$ the$_{10}$ book$_{11}$
(H5) he$_8$ read$_9$ the$_{10}$ book$_{11}$ because$_7$ he$_1$ was$_2$ interested$_3$ in$_4$ world$_5$ history$_6$

H5's word order is [8, 9, 10, 11, 7, 1, 2, 3, 4, 5, 6]. The number of increasing pairs is: ${}_4C_2 = 6$ pairs in [8, 9, 10, 11] and ${}_6C_2 = 15$ pairs in [1, 2, 3, 4, 5, 6]. Then we obtain $\tau = 2 \times (6 + 15)/{}_{11}C_2 - 1 = -0.236$. On the other hand, $\sum_i d_i^2 = 5^2 \times 6 + 2^2 + 7^2 \times 4 = 350$, and we obtain $\rho = 1 - 350/{}_{12}C_3 = -0.591$.

Therefore, both Spearman's ρ and Kendall's τ give very bad scores to the misleading translation H0. This fact implies they are much better metrics than BLEU, which gave a good score to it. ρ is much lower than τ as we expected.

In general, we can use higher-order n-grams for this alignment, but here we use only unigrams and bigrams for simplicity. This algnment algorithm is given in Figure 1. Since some hypothesis words do not have corresponding reference words, the output integer list `worder` is sometimes shorter than the evaluated sentence. Therefore, we should not use $\texttt{worder}[i] - i$ as d_i directly. We have to renumber the list by rank as we did in Section 1.

```
Read a hypothesis sentence $h = h_1 h_2 \ldots h_m$
and its reference sentence $r = r_1 r_2 \ldots r_n$.
Initialize worder with an empty list.
For each word $h_i$ in $h$:
    • If $h_i$ appears only once each in $h$ and $r$, append $j$
      s.t. $r_j = h_i$ to worder.
    • Otherwise, if the bigram $h_i h_{i+1}$ appears only once
      each in $h$ and $r$, append $j$ s.t. $r_j r_{j+1} = h_i h_{i+1}$ to
      worder.
    • Otherwise, if the bigram $h_{i-1} h_i$ appears only once
      each in $h$ and $r$, append $j$ s.t. $r_{j-1} r_j = h_{i-1} h_i$ to
      worder.
Return worder.
```

Figure 1: Word alignment algorithm for rank correlation

2.2 Word order metrics and meta-evaluation metrics

These rank correlation metrics sometimes have negative values. In order to make them just like other automatic evaluation metrics, we normalize them as follows.

- Normalized Kendall's τ: NKT = $(\tau + 1)/2$.
- Normalized Spearman's ρ: NSR = $(\rho + 1)/2$.

Accordingly, NKT is 0.382 and NSR is 0.205.

These metrics are defined only when the number of aligned words is two or more. We define both NKT and NSR as zero when the number is one or less. Consequently, these normalized metrics have the same range [0, 1].

In order to avoid confusion, we use these abbreviations (NKT and NSR) when we use rank correlations as **word order metrics**, because these correlation metrics are also used in the machine translation community for **meta-evaluation**. For meta-evaluation, we use Spearman's ρ and Pearson's correlation coefficient and call them "Spearman" and "Pearson," respectively.

2.3 Overestimation problem

Since we measure the rank correlation of only corresponding words, these metrics will overestimate the correlation. For instance, a hypothesis sentence might have only two corresponding words among

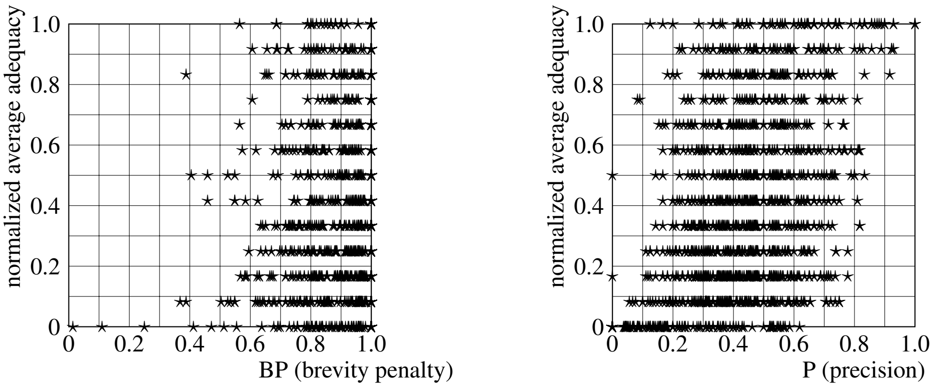


Figure 2: Scatter plots of normalized average adequacy with brevity penalty (left) and precision (right). (Each ⋆ corresponds to one sentence generated by one MT system)

dozens of words. In this case, these two words determine the score of the whole sentence. If the two words appear in their order in the reference, the whole sentence obtains the best score, NSR = NKT = 1.0, in spite of the fact that only two words matched.

Solving this overestimation problem is the second objective of this paper. BLEU uses "Brevity Penalty (BP)" (Section 1) to reduce the scores of too-short sentences. We can combine the above word order metrics with BP, e.g., NKT $\times$ BP and NSR $\times$ BP.

However, we cannot very much expect from this solution because BP scores do not correlate with human judgments well. The left graph of Figure 2 shows a scatter plot of BP and "normalized average adequacy." This graph has 15 (systems) $\times$ 100 (sentences) dots. Each dot ($\star$) corresponds to one sentence from one translation system.

In the NTCIR-7 data, three human judges gave five-point scores (1, 2, 3, 4, 5) for "adequacy" and "fluency" of each translated sentence. Although each system translated 1,381 sentences, only 100 sentences were evaluated by the judges.

For each translated sentence, we averaged three judges' adequacy scores and normalized this average x by $(x-1)/4$. This is our "normalized average adequacy," and the dots appears only at multiples of $1/3 \times 1/4$.

This graph shows that BP has very little correlation with adequacy, and we cannot expect BP to improve the meta-evaluation performance very much. Perhaps, BP's poor performance was caused by the fact that most MT systems output almost the same number of words, and if the number exceeds the length of the reference, BP=1.0 holds.

Therefore, we have to consider other modifiers for this overestimation problem. We can use other common metrics such as precision, recall, and F-measure to reduce the overestimation of NSR and NKT.

- Precision: $P = c/|h|$, where c is the number of corresponding words and $|h|$ is the number of words in the hypothesis sentence h.
- Recall: $R = c/r$, where $|r|$ is the number of words in the reference sentence r.
- F-measure: $F_\beta = (1+\beta^2)PR/(\beta^2 P + R)$, where β is a parameter.

In (R2)&(H2)'s case, precision is 5/7 = 0.714 and recall is 5/5 = 1.000.

Which metric should we use? Our preliminary experiments with NTCIR-7 data showed that **precision correlated best with adequacy** among these three metrics (P, R, and $F_{\beta=1}$). In addition, BLEU is essentially made for precision. Therefore, precision seems the most promising modifier.

The right graph of Figure 2 shows a scatter plot of precision and normalized average adequacy. The graph shows that precision has more correlation with adequacy than BP. We can observe that sentences with very small P values usually obtain very low adequacy scores but those with mediocre P values often obtain good adequacy scores.

If we multiply P directly by NSR or NKT, those sentences with mediocre P values will lose too much of their scores. The use of $\sqrt{x}$ will mitigate this problem. Since $\sqrt{P}$ is closer to 1.0 than P itself, multiplication of $\sqrt{P}$ instead of P itself will save these sentences. If we apply $\sqrt{x}$ twice ($\sqrt{\sqrt{P}} = \sqrt[4]{P}$), it will further save them. Therefore, we expect $\times\sqrt{P}$ and $\times\sqrt[4]{P}$ to work better than $\times P$. Now, we propose two new metrics:

$$\mathrm{NSR}P^{\alpha} \quad \text{and} \quad \mathrm{NKT}P^{\alpha},$$

where α is a parameter ($0 \leq \alpha \leq 1$).

3 Experiments

3.1 Meta-evaluation with NTCIR-7 data

In order to compare automatic translation evaluation methods, we use submissions to the NTCIR-7 Patent Translation (PATMT) task (Fujii et al., 2008). Fourteen MT systems participated in the Japanese-English intrinsic evaluation. There were two Rule-Based MT (RMBT) systems and one Example-based MT (EBMT) system. All other systems were Statistical MT (SMT) systems. The task organizers provided a baseline SMT system. These 15 systems translated 1,381 Japanese sentences into English. The organizers evaluated these translations by using BLEU and human judgments. In the human judgements, three experts independently evaluated 100 selected sentences in terms of '*adequacy*' and '*fluency*.'

For automatic evaluation, we used a single reference sentence for each of these 100 manually evaluated sentences. Echizen-ya et al. (2009) used multi-reference data, but their data is not publicly available yet.

For this meta-evaluation, we measured the *corpus-level* correlation between the human evaluation scores and the automatic evaluation scores. We simply averaged scores of 100 sentences for the proposed metrics. For existing metrics such as BLEU, we followed their definitions for corpus-level evaluation instead of simple averages of sentence-level scores. We used default settings for conventional metrics, but we tuned GTM (Melamed et al., 2007) with `-e` option. This option controls preferences on longer word runs. We also used the paraphrase database TERp (`http://www.umiacs.umd.edu/~snover/terp`) for METEOR (Banerjee and Lavie, 2005).

3.2 Meta-evaluation with WMT-07 data

We developed our metric mainly for automatic evaluation of translation quality for distant language pairs such as Japanese-English, but we also want to know how well the metric works for similar language pairs. Therefore, we also use the WMT-07 data (Callison-Burch et al., 2007) that covers only European language pairs. Callison-Burch et al. (2007) tried different human evaluation methods and showed detailed evaluation scores. The Europarl test set has 2,000 sentences, and The News Commentary test set has 2,007 sentences.

This data has different language pairs: Spanish, French, German ⇒ English. We exclude Czech-English because there were so few systems (See the footnote of p. 146 in their paper).

4 Results

4.1 Meta-evaluation with NTCIR-7 data

Table 1 shows the main results of this paper. The left part has corpus-level meta-evaluation with **adequacy**. Error metrics, WER, PER, and TER, have negative correlation coefficients, but we did not show their minus signs here.

Both NSR-based metrics and NKT-based metrics perform better than conventional metrics for this NTCIR PATMT JE translation data. As we expected, $\times$BP and $\times P^{(1/1)}$ performed badly. Spearman of BP itself is zero.

NKT performed slightly better than NSR. Perhaps, NSR penalized alternative good translations too much. However, one of the NSR-based metrics, NSR$P^{1/4}$, gave the **best Spearman score of 0.947**, and the difference between NSRP^{α} and NKTP^{α} was small. Modification with P led to this improvement.

NKT gave the best Pearson score of 0.922. However, Pearson measures linearity and we can change its score through a nonlinear monotonic function without changing Spearman very much. For instance, (NSR$P^{1/4})^{1.5}$ also has **Spearman of 0.947** but its **Pearson is 0.931**, which is better than NKT's 0.922. Thus, we think Spearman is a better **meta-**evaluation metric than Pearson.

Table 1: NTCIR-7 Meta-evaluation: correlation with human judgments (Spm = Spearman, Prs = Pearson)

human judge	**Adequacy**		Fluency	
eval\ meta-eval	Spm	Prs	Spm	Prs
P	0.615	0.704	0.672	0.876
R	0.436	0.669	0.461	0.854
$F_{\beta=1}$	0.525	0.692	0.543	0.871
BP	0.000	0.515	-0.007	0.742
NSR	0.904	0.906	0.869	0.910
NSR$P^{1/8}$	0.937	0.905	0.890	0.934
NSR$P^{1/4}$	**0.947**	0.900	0.901	0.944
NSR$P^{1/2}$	0.937	0.890	0.926	0.949
NSR$P^{1/1}$	0.883	0.872	0.883	0.939
NSR × BP	0.851	0.874	0.769	0.910
NKT	0.940	0.922	0.887	0.931
NKT$P^{1/8}$	0.940	0.913	0.908	0.944
NKT$P^{1/4}$	0.940	0.904	0.908	0.949
NKT$P^{1/2}$	0.929	0.890	0.897	0.949
NKT$P^{1/1}$	0.897	0.869	0.879	0.936
NKT × BP	0.829	0.878	0.726	0.918
ROUGE-L	0.903	0.874	0.889	0.932
ROUGE-S(4)	0.593	0.757	0.640	0.869
IMPACT	0.797	0.813	0.751	0.932
WER	0.894	0.822	0.836	0.926
TER	0.854	0.806	0.372	0.856
PER	0.375	0.642	0.393	0.842
METEOR(TERp)	0.490	0.708	0.508	0.878
GTM(-e 12)	0.618	0.723	0.601	0.850
NIST	0.343	0.661	0.372	0.856
BLEU	0.515	0.653	0.500	0.795

The right part of Table 1 shows correlation with fluency, but **adequacy is more important**, because our motivation is to provide a metric that is useful to reduce incomprehensible or misunderstanding outputs of MT systems. Again, the correlation-based metrics gave better scores than conventional metrics, and BP performed badly. NSR-based metrics proved to be as good as NKT-based metrics.

Meta-evaluation scores of the de facto standard BLEU is much lower than those of other metrics. Echizen-ya et al. (2009) reported that IMPACT performed very well for *sentence-level* evaluation of NTCIR-7 PATMT JE data. This *corpus-level* result also shows that IMPACT works better than BLEU, but ROUGE-L, WER, and our methods give better scores than IMPACT.

Table 2: WMT-07 meta-evaluation: Each source language has two columns: the left one is News Corpus and the right one is Europarl.

Spearman's ρ with human "rank"						
source	French		Spanish		German	
NSR	0.775	0.837	0.523	0.766	0.700	0.593
NSR$P^{1/8}$	0.821	0.857	0.786	0.595	0.400	0.685
NSR$P^{1/4}$	0.821	0.857	0.786	0.455	0.400	0.714
NSR$P^{1/2}$	0.821	0.857	0.786	0.347	0.400	0.714
NKT	0.845	0.857	0.607	0.838	0.700	0.630
NKT$P^{1/8}$	0.793	0.857	0.786	0.595	0.400	0.714
NKT$P^{1/4}$	0.793	0.857	0.786	0.524	0.400	0.714
NKT$P^{1/2}$	0.793	0.857	0.786	0.347	0.400	0.714
BLEU	0.786	0.679	0.750	0.595	0.400	0.821
WER	0.607	0.857	0.750	0.429	0.000	0.500
ROUGEL	0.893	0.739	0.786	0.707	0.700	0.857
ROUGES	0.883	0.679	0.786	0.690	0.400	0.929

4.2 Meta-evaluation with WMT-07 data

Callison-Burch et al. (2007) have performed different human evaluation methods for different language pairs and different corpora. Their Table 5 shows inter-annotator agreements for the human evaluation methods. According to their table, the "sentence ranking" (or "rank") method obtained better agreement than "adequacy." Therefore, we show Spearman's ρ for "rank." We used the scores given in their Tables 9, 10, and 11. (The "constituent" methods obtained the best inter-annotator agreement, but these methods focus on local translation quality and have nothing to do with global word order, which we are discussing here.)

Table 2 shows that our metrics designed for distant language pairs are comparable to conventional methods even for similar language pairs, but ROUGE-L and ROUGE-S performed better than ours for French News Corpus and German Europarl. BLEU scores in this table agree with those in Table 17 of Callison-Burch et al. (2007) within rounding errors.

After some experiments, we noticed that the use of R instead of P often gives better scores for WMT-07, but it degrades NTCIR-7 scores. We can extend our metric by F_β, weighted harmonic mean of P and R, or any other interpolation, but the introduction of new parameters into our metric makes it difficult to control. Improvement without new parameters is beyond the scope of this paper.

5 Discussion

It has come to our attention that Birch et al. (2010) has independently proposed an automatic evaluation method based on Kendall's τ. First, they started with Kendall's τ distance, which can be written as "$1 - \mathrm{NKT}$" in our terminology, and then subtracted it from one. Thus, their metric is nothing but NKT.

Then, they proposed application of the square root to get better Pearson by improving "the sensitivity to small reorderings." Since they used "Kendall's τ" and "Kendall's τ distance" interchangeably, it is not clear what they mean by "$\sqrt{\ }$ Kendall's τ," but perhaps they mean $1 - \sqrt{1 - \mathrm{NKT}}$ because $\sqrt{\mathrm{NKT}}$ is more insensitive to small reorderings. Table 3 shows the performance of these metrics for NTCIR-7 data. Pearson's correlation coefficient with adequacy was improved by $1 - \sqrt{1 - \mathrm{NKT}}$, but other scores were degraded in this experiment.

The difference between our method and Birch et al. (2010)'s method comes from the fact that we used Japanese-English translation data and Spearman's correlation for meta-evaluation, whereas they used Chinese-English translation data and only Pearson's correlation for meta-evaluation. Chinese word order is different from English, but Chinese is a Subject-Verb-Object (SVO) language and thus is much closer to English word order than Japanese, a typical SOV language.

We preferred NSR because it penalizes global word order mistakes much more than does NKT, and as discussed above, global word order mistakes often lead to incomprehensibility and misunderstanding.

On the other hand, they also tried Hamming distance, and summarized their experiments as follows:

> However, the Hamming distance seems to be more informative than Kendall's tau for small amounts of reordering.

This sentence and the introduction of the square root to NKT imply that Chinese word order is close to that of English, and they have to measure subtle word order mistakes.

Table 3: NTCIR-7 meta-evaluation: Effects of square root ($b(x) = 1 - \sqrt{1 - x}$)

	NKT	$\sqrt{\mathrm{NKT}}$	b(NKT)
Spearman w/ adequacy	0.940	0.940	0.922
Pearson w/ adequacy	0.922	0.817	0.941
Spearman w/ fluency	0.887	0.865	0.858
Pearson w/ fluency	0.931	0.917	0.833

In spite of these differences, the two groups independently recognized the usefulness of rank correlations for automatic evaluation of translation quality for distant language pairs.

In their WMT-2010 paper (Birch and Osborne, 2010), they multiplied NKT with the brevity penalty and interpolated it with BLEU for the WMT-2010 shared task. This fact implies that incomprehensible or misleading word order mistakes are rare in translation among European languages.

6 Conclusions

When Statistical Machine Translation is applied to distant language pairs such as Japanese and English, word order becomes an important problem. SMT systems often fail to find an appropriate translation because of a large search space. Therefore, they often output misleading or incomprehensible sentences such as "A because B" vs. "B because A." To penalize such inadequate translations, we presented an automatic evaluation method based on rank correlation. There were two questions for this approach. First, which correlation coefficient should we use: Spearman's ρ or Kendall's τ? Second, how should we solve the overestimation problem caused by the nature of one-to-one correspondence?

We answered these questions through our experiments using the NTCIR-7 PATMT JE translation data. For the first question, τ was slightly better than ρ, but ρ was improved by precision. For the second question, it turned out that BLEU's Brevity Penalty was counter-productive. A precision-based penalty gave a better solution. With this precision-based penalty, both ρ and τ worked well and they outperformed conventional methods for NTCIR-7 data. For similar language pairs, our method was comparable to conventional evaluation methods. Future work includes extension of the method so that it can outperform conventional methods even for similar language pairs.

References

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for MT evaluation with improved correlation with human judgements. In *Proc. of ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and Summarization*, pages 65–72.

Alexandra Birch and Miles Osborne. 2010. LRscore for evaluating lexical and reordering quality in MT. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 327–332.

Alexandra Birch, Miles Osborne, and Phil Blunsom. 2010. Metrics for MT evaluation: evaluating reordering. *Machine Translation*, 24(1):15–26.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluatiing the role of Bleu in machine translation research. In *Proc. of the Conference of the European Chapter of the Association for Computational Linguistics*, pages 249–256.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Chrstof Monz, and Josh Schroeder. 2007. (Meta-)Evaluation of machine translation. In *Proc. of the Workshop on Machine Translation (WMT)*, pages 136–158.

Etienne Denoual and Yves Lepage. 2005. BLEU in characters: towards automatic MT evaluation in languages without word delimiters. In *Companion Volume to the Proceedings of the Second International Joint Conference on Natural Language Processing*, pages 81–86.

Hiroshi Echizen-ya and Kenji Araki. 2007. Automatic evaluation of machine translation based on recursive acquisition of an intuitive common parts continuum. In *Proceedings of MT Summit XII Workshop on Patent Translation*, pages 151–158.

Hiroshi Echizen-ya, Terumasa Ehara, Sayori Shimohata, Atsushi Fujii, Masao Utiyama, Mikio Yamamoto, Takehito Utsuro, and Noriko Kando. 2009. Meta-evaluation of automatic evaluation methods for machine translation using patent translation data in ntcir-7. In *Proceedings of the 3rd Workshop on Patent Translation*, pages 9–16.

Atsushi Fujii, Masao Utiyama, Mikio Yamamoto, and Takehito Utsuro. 2008. Overview of the patent translation task at the NTCIR-7 workshop. In *Working Notes of the NTCIR Workshop Meeting (NTCIR)*, pages 389–400.

Hideki Isozaki, Katsuhito Sudoh, Hajime Tsukada, and Kevin Duh. 2010. Head Finalization: A simple reordering rule for SOV languages. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 250–257.

Maurice G. Kendall. 1975. *Rank Correlation Methods*. Charles Griffin.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proc. of the North American Chapter of the Association of Computational Linguistics (NAACL)*, pages 71–78.

Dan Melamed, Ryan Green, and Joseph P. Turian. 2007. Precision and recall of machine translation. In *Proc. of NAACL-HLT*, pages 61–63.

Kishore Papineni, Salim Roukos, Todd Ward, John Henderson, and Florence Reeder. 2002a. Corpus-based comprehensive and diagnostic MT evaluation: Initial Arabic, Chinese, French, and Spanish Results. In *Proc. of the International Conference on Human Language Technology Research (HLT)*, pages 132–136.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002b. BLEU: a method for automatic evaluation of machine translation. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 311–318.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*.

An approach for generating personalized views from normalized electronic dictionaries : A practical experiment on Arabic language

Aida Khemakhem
MIRACL Laboratory
FSEGS, B.P. 1088,
3018 Sfax, Tunisia
khemakhem.aida@gmail.com

Bilel Gargouri
MIRACL Laboratory
FSEGS, B.P. 1088,
3018 Sfax, Tunisia
bilel.gargouri@fsegs.rnu.tn

Abdelmajid Ben Hamadou
MIRACL Laboratory
ISIMS, Cité Ons,
3021 Sfax, Tunisia
abdelmajid.benhamadou@isimsf.rnu.tn

Abstract

Electronic dictionaries covering all natural language levels are very relevant for the human use as well as for the automatic processing use, namely those constructed with respect to international standards. Such dictionaries are characterized by a complex structure and an important access time when using a querying system. However, the need of a user is generally limited to a part of such a dictionary according to his domain and expertise level which corresponds to a specialized dictionary. Given the importance of managing a unified dictionary and considering the personalized needs of users, we propose an approach for generating personalized views starting from a normalized dictionary with respect to Lexical Markup Framework LMF-ISO 24613 norm. This approach provides the re-use of already defined views for a community of users by managing their profiles information and promoting the materialization of the generated views. It is composed of four main steps: (i) the projection of data categories controlled by a set of constraints (related to the user's profiles), (ii) the selection of values with consistency checking, (iii) the automatic generation of the query's model and finally, (iv) the refinement of the view. The proposed approach was consolidated by carrying out an experiment on an LMF normalized Arabic dictionary.

1 Introduction

Electronic dictionaries are very useful in nowadays society, with the globalization and the increase of world communication and exchanges. There are clearly identified needs of dictionaries for human use as well as for automatic processing use.

Given the importance of having recourse to standards when constructing such lexical resources in order to promote the reuse and the fusion, the standardization committee ISO TC37/SC4 has recently validated the Lexical Markup Framework norm (LMF) project under the standard ISO 24 613 (Francopoulo and George 2008). LMF provides a common and shared representation of lexical objects that allows for the encoding of rich linguistic information, including among others morphological, syntactic, and semantic aspects. The LMF proposal is distinguished by the separate management of the hierarchical data structure (meta-model) and elementary linguistic descriptors (data categories) which promotes to cover several languages.

A normalized dictionary covers wide areas that include all lexical information of a given language and which are useful both for human use and for Natural Language Processing use in accordance with the kind of the user (linguist, lexicographer, developer, etc.), the level of the user (learner, expert, etc.) and the domain of the use (linguistic, medicine, biology, etc.). These dictionaries are characterized by a complex structure that supports the richness of natural languages. Therefore, dealing with a unique and complete dictionary is well for the manage task. However, such dictionaries are large and can be time consuming when querying their contents especially on the web. Moreover, displaying all details when some of which are not useful for the field of the query research is a

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 953–960,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

nuisance for the user. So, it will be interesting to reduce the displayed details according to the domain or to the expertise level of the user by generating personalized views (virtual or materialized) in order to appropriate the use of such dictionaries to the user needs.

The idea of creating document views is not a new concept but applying it on LMF normalized dictionaries is a new one. Indeed, it has been some attempts for dictionary creating in accordance to the TEI consortium (Véronis and Ide 1996) but the problem was the fact that created textual views (corresponding to the surface structure) or data base views (corresponding to the deep structure) were not customized. Others propositions are very interesting but they concern the ontology domain when dealing with the concept of point of view (Corby and al 2005).

In this paper, we propose an approach that favors the use of normalized dictionaries by generating virtual/materialized personalized views. This approach is based on the profiles information associated to user's community which helps to retrieve already defined views stored in a library of views. For the illustration, we use an Arabic LMF normalized dictionary (Baccar and al. 2008, Khemakhem and al., 2009) developed in the framework of an Arabic project supervised by the ALECSO (The Arab League Educational, Cultural and Scientific Organization) and founded by the University of King Abdul-Aziz in the Kingdom of Saudi Arabia[1]. An environment supporting the proposed approach was implemented. At present, it concerns the Arabic language.

The present paper is outlined as follows. We will start by giving an overview of projects that use LMF notably for the construction and the exploitation of electronic dictionaries. Then, we will present the foundation of the proposed approach related to the profile and the view concepts. After that, we will explain the different steps of the view's generating approach. Finally, we will bring back the experimentation that we carried out on a normalized Arabic dictionary using the proposed approach.

2 State of the art of projects using LMF

After the emergence of LMF, some projects were launched in order to construct or exploit electronic dictionaries in accordance with this norm. Among others we note LEXUS (Kirsch 2005) (Kemps-Snijders and al 2006), LIRICS (LIRICS 2005) and LMF-QL (Ben Abderrahmen and al 2007). All these labors have been recourse to Web service technology that favors to invoke, locally or afar, appropriate services for the management or the exploitation of a normalized dictionary.

LEXUS offers an interface permitting to the user to define formats of lexical bases with a perspective to enable the construction of lexical bases according to the LMF model. However, it does not allow the verification of the compliance between the model and the norm.

LIRICS proposes some APIs that focus especially on the management of lexical data base. These APIs offer the possibility to work on the structure of the LMF base by adding, modifying or deleting components of LMF model. However, there is no interface which facilitates the use of these APIs.

LMF-QL provides Web services to be used while developing lingware systems. These services offer the exploitation of a normalized lexical base without having any piece of information about its content and its structure. The results of these services may be personalized dictionaries given in an XML format. However, it covers only the morphological level.

Concerning the construction of personalized dictionaries using the works mentioned above, we can notice that the user must have an idea about the content of the desired dictionary and its structure. He must also have acquaintances with queries generation to satisfy his requirements.

Finally, we note an absence of works dealing with the generation of views starting from LMF standardized dictionary.

3 Foundation of the approach

An electronic dictionary can be used by many users who have different requirements. Indeed, by being a language learner, a researcher in linguistics or a teacher's, needs and uses are not the same. Therefore, it will be better to have a tool (editor) allowing generating a suitable view. The making of a view of the dictionary might be difficult for some kinds of users, so the recourse to user profiles may facilitate this task. One can note that the user profile is very important to guide the user through the retrieval and the reuse of existent views corresponding to his profile.

[1] www.almuajam.org

3.1 A user profile definition

Generally, all features characterizing a user or a group of users can be grouped under the term of a user profile. For electronic dictionaries, a user profile is a dataset that concerns a community of users of the dictionary.

Every profile is characterized by a category, a level of expertise and a field. Indeed, we classify the views of the dictionary according to a profile that is based on a selection of these three criteria. The formal representation of a profile is the following:

P : < K, L, F>

K: the kind of user: lexicographer, linguist, lingware system developer, etc.

L: the level of the expertise: beginner, student, expert, etc.

F: the field of user: medicine, sport, biology, general, etc.

3.2 A View definition

A view of a dictionary is a synthesis of a dictionary interrogation query. We can consider it as a specialized or lexical dictionary, supported by a query.

A dictionary view allows to filter some lexicographic information and to hide others that are useless for some users.

The formal representation of a view:

View : < D, P, C >

D: dictionary: each view is specific to a normalized dictionary.

P: profile of the view (see previous section).

C: it is a set of properties which characterizes the model of the view. Each property has the following representation:

C : <A, V, W>

A: attribute is a simple representation of a characteristic model. This characteristic may be a class (Lemma, Sense…), a feature (definition, pos, genre,…) or a relationship (RelatedForm, SenseRelation…) as indicated in Figure 3.

V: value: each attribute can have a set of values. For example, the values verb and noun for the attribute POS (Part Of Speech).

W: weight of a property. It may take the values 0 or 1.

- If the weight equals to 0, then this property is mandatory only for part of the lexical entry.
- If the weight equals to 1, then this property is mandatory for each lexical entry of this view.

3.3 Different types of views

There are two types of views:

- Virtual view: the results of this view are calculated upon request. In this case, interrogating queries of this view might generate a composed query that interrogates directly the principal dictionary (underlying).
- Materialized view (physical): a physical copy faithful to the view definition which is stored and maintained.

4 Proposed approach

In this section we describe the proposed approach through the detail that we will give for each one of its four steps. These steps are illustrated in Figure 1.

- Projection of the view model: includes the specification of data categories (linguistic information), controlled by constraints in order to build a suitable normalized and valid model. It can be started by already existing profiles.
- Selection and checkout of the coherence: concerns the specification of some values for data categories (DC) already specified. We use coherence rules to check the constancy knowing that there are strong dependencies between some DCs and values of other DCs (see section 4.2).
- Automatic generation of the model and the query: includes the model refinement and the checking of constraints by priority.
- Refinement of the view: involves the validation of a new view of dictionary, by adding the elements that are related to the lexical entries of this view.

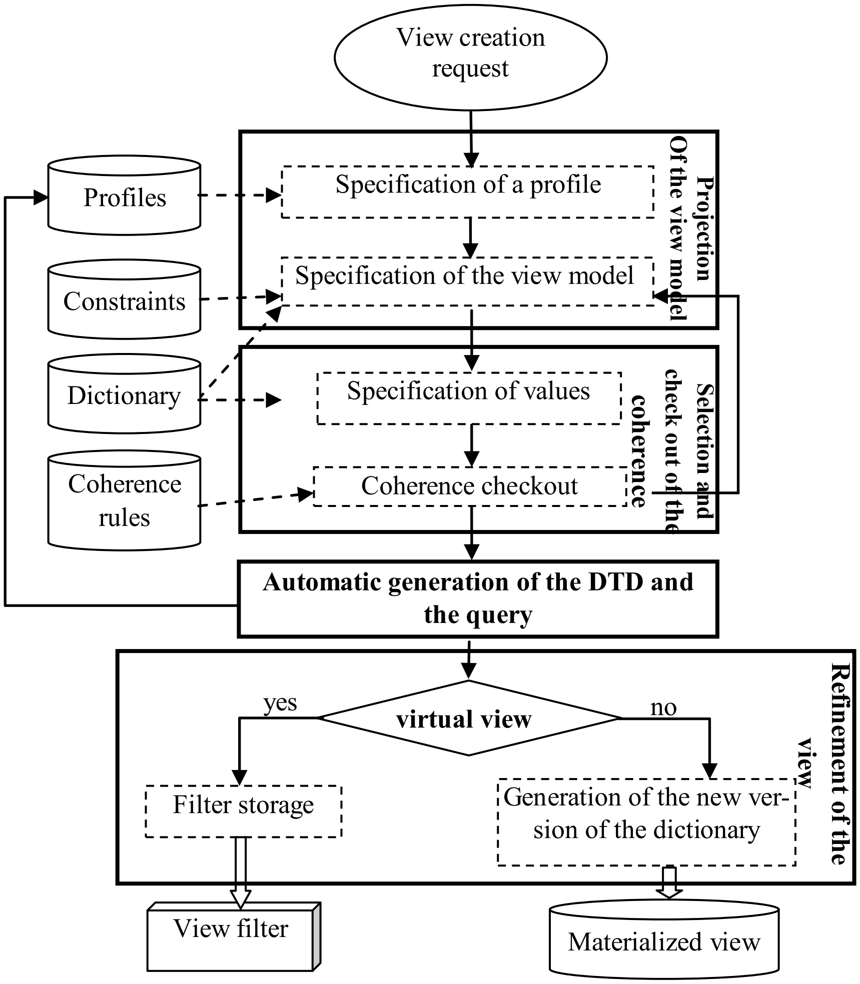

View creation request
Profiles
Constraints
Dictionary
Coherence rules
Specification of a profile
Specification of the view model
Projection Of the view model
Specification of values
Coherence checkout
Selection and check out of the coherence
Automatic generation of the DTD and the query
virtual view
yes
no
Filter storage
Generation of the new version of the dictionary
Refinement of the view
View filter
Materialized view

Figure 1.The approach for generating personalized dictionary views.

4.1 Projection of the view model

The UML model of the dictionary is difficult to understand. So, we suggest a simpler representation which is more abstract. The choice can be started by already existing profiles in order to avoid views redundancy by the reuse of the previous ones and help users.

a. The specification of a profile

A user profile is a description that corresponds to a user community. We use the features of profile to filter the DCs. Indeed, we offer to the user only DCs that correspond to its field and its level based on the weight assigned to each DC. We assigned these weights according to a study on the needs of each user level. This study is based on the specific documents and dictionaries for each user level. By example for beginner level, we studied the school books to extract the information (root, schema,…) needed at this level.

It also facilitates and accelerates the task of needs specification and permits to avoid views redundancy. The projection phase is started by the specification of the user profile that requires the choice of its category, its level and its field. Then, if the user wants to consult the previous views of his profile, we display all the views associated to this profile. Otherwise, we offer the DCs specific to its field and its level.

b. Constraints

The abstraction of the model can hides relations between DCs. Indeed, during their specification there is a risk of having views with a non valid model. So, the DC specification must be controlled by constraints rules such as:

- If we select the field, the semantic class or the nature then the definition must be selected.
- If the relation between syntactic behaviors and senses is selected then the definition must be selected.

4.2 Selection and checkout of the coherence

Most of data categories use a list of values such as part-of-speech (pos), scheme, field, etc. Values specification of some DCs might influence the presence or the absence of the other DCs. For example, if the user has chosen DCs: pos, root, scheme, gender and number; then, he has fixed the value du pos =”particle” (it means that he needs only particles) and the value of the number =”singular”. In this case, we note an incoherence problem since the DC “number” is among particles characteristics but it concerns nouns. In this case, we must request the user to rectify the specifications. The selection must contain a checkout phase of coherence of DCs specified with the already existing data in the dictionary. This phase is based on coherence rules which ensure consistency between DCs and the specified values.

4.3 Automatic generation of the DTD and the query

We use the Document Type Definition (DTD) of the LMF norm and DC specifications of the model to automatically generate the DTD of the view. We use algorithms to generate DTD elements, respecting the order of the participating classes and ensuring classes relations.

The automatic generation of the query (i.e. using XQuery) involves two steps: the first one concern the specification of conditions for the selection of a lexical entry and its related information (i.e., semantic, syntactic). The second step permits the definition of an XML representation of a lexical entry. This step is based on the projection specified by the user and the priority order of DCs. There are DCs that influence the presence of the lexical entry and others that only influence the presence of the sense.

4.4 Refinement of the view

The steps of this phase depend on the type of the view. If it’s a virtual one, then it’s necessary to save the query already generated. This query will be used during the operation of this view. If it’s a materialized one, then query results are a part of the dictionary and must be saved for the operation of the view. The second case may give us a non valid XML base, especially when there are two lexical entries in relation and our query will select one of them that have an identifier of the other entry that is not selected. We recall that the lexical entries may have morphological links with other lexical entries and semantic links with other senses.

Indeed, after recording query‘s results, we move to the step of refinement. This step consists to valid the new personalized dictionary, adding lexical entries, senses and syntactic frame in relation with these results.

5 Experimentation

5.1 The normalized Arabic dictionary

In order to experiment our approach, we are going to use the normalized Arabic interactive dictionary containing more than 38000 lexical entries and developed in the framework of an Arabic project[2] supervised by the ALECSO. This dictionary is modeled according to the meta-model proposed by LMF[3] (ISO 24613) and uses data categories generated by the DCR[4] norm (ISO 12620). The dictionary pattern is composed of classes selected from the kernel or from one of its extensions (morphological, semantic, syntactic, MRD) in order to see a dictionary covering most of new dictionary’s needs (Baccar and al 2008). Since there are many information that can be classified in multi extensions in the same time, the norm’s editor have chosen to put them in one of these extensions. For this reason, we did not use only Machine Readable Dictionary (MRD) extension. This pattern valorizes derivation phenomenon in Arabic language and neutralizes the differences between lexicographical schools, ensuring language evolution. In fact, we have considered

[2] www.almuajam.org/

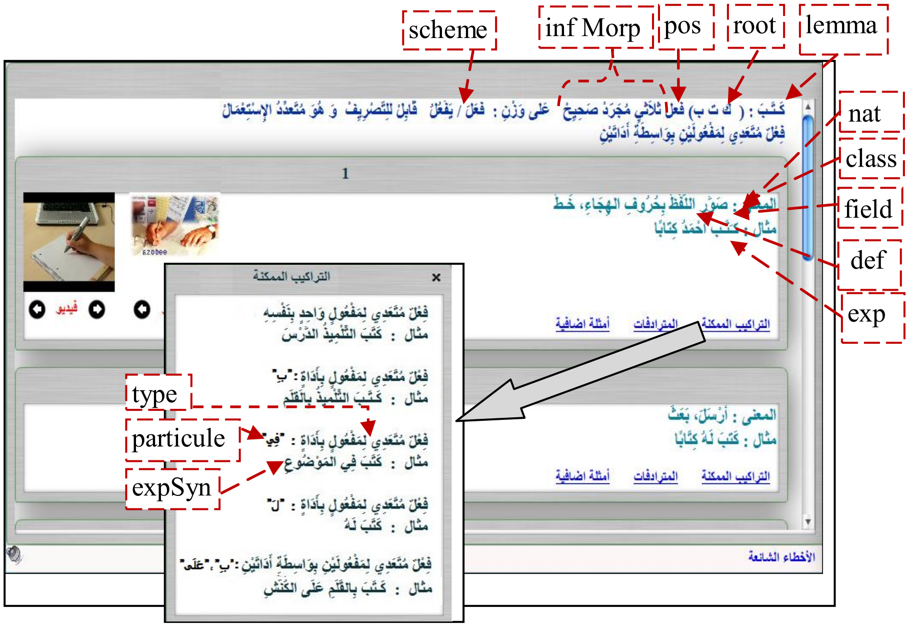
[3] www.lexicalmarkupframework.org/

[4] www.isocat.org/

roots (ك ت ب “k t b”), derived forms (كَتَبَ “kataba” (write), كَاتِب ”kâtib” (writer)), invariable words (إنَّ ”inna” (indeed), حَتَّى “hattâ” (in order)) and non-Arab origin words (كَمْبِيُوتَر “computer”, أَنْتَرْنَات “internet”) as lexical entries that can have morphological relations (i.e., RelatedForm relation).

In addition, this dictionary is rich with semantic information (i.e., definitions, examples, subject field, semantic class) and syntactic information (i.e., subcategorisation frame). It ensures the link between senses and their possible syntactic behaviors.

The Figure 2, given below, shows a part of the lexical entry “كَتَبَ” ”kataba” (write) which gives an idea about the structure of this dictionary.



كَتَبَ : (كَ ت ب) فعلٌ ثلاثي مُجَرَّدٌ صَحِيحٌ على وَزْنِ : فَعَلَ / يَفعُلُ قابِلٌ للتَّصرِيفِ و هُوَ مُتَعَدِّدُ الإستِعمَالِ
فِعْلٌ مُتَعَدِي لِمَفْعُولَيْنِ بِوَاسِطَةِ أَدَاتَيْنِ

1

المعنى : صَوَّرَ اللَّفظَ بِحُرُوفِ الهِجَاءِ، خَطَّ
مثال : كَتَبَ أحمَدُ كِتَابًا

التراكيب الممكنة المترادفات أمثلة اضافية

التراكيب الممكنة ×

فِعْلٌ مُتَعَدِي لِمَفْعُولٍ وَاحِدٍ بِنَفْسِهِ
مثال : كَتَبَ التِّلْمِيذُ الدَّرْسَ

فِعْلٌ مُتَعَدِي لِمَفْعُولٍ بِأَدَاةٍ :"بِ"
مثال : كَتَبَ التِّلْمِيذُ بِالْقَلَمِ

فِعْلٌ مُتَعَدِي لِمَفْعُولٍ بِأَدَاةٍ : "فِي"
مثال : كَتَبَ فِي المَوْضُوعِ

فِعْلٌ مُتَعَدِي لِمَفْعُولٍ بِأَدَاةٍ : "لَ"
مثال : كَتَبَ لَهُ

فِعْلٌ مُتَعَدِي لِمَفْعُولَيْنِ بِوَاسِطَةِ أَدَاتَيْنِ :"بِ"،"عَلَى"
مثال : كَتَبَ بِالقَلَمِ عَلَى الكُنَّشِ

المعنى : أرْسَلَ، بَعَثَ
مثال : كَتَبَ لَهُ كِتَابًا

التراكيب الممكنة المترادفات أمثلة اضافية

الأخطاء الشائعة

pos : part of speech
nat : nature
class : class
field : field
exp : example for a sense
inf Morp : Morphlogical piece of information
def : definition
expSyn : example of using a type
type : type of a syntactic frame

Figure 2. Example of a lexical entry of the normalized Arabic dictionary.

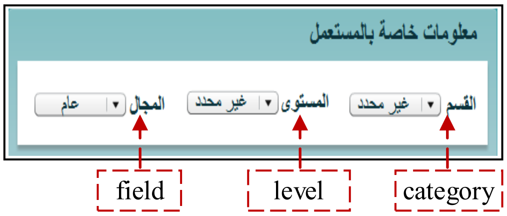
In this Figure, we highlight some properties of the used dictionary such as:

- The diversity of information: morphology, semantics, syntax, image, video, etc.
- The sense relations (i.e., synonym) link two senses and not two lexical entries
- The precision of the syntactic behavior. Indeed, each syntactic behavior has a type; the particles needed an example and its definition.
- The structure of a lexical entry varies according to its part of speech

5.2 Experiment of the approach

We illustrate in this section the generation process of a personalized dictionary view. We have setup a computing system online, that allows the user to make his view of the dictionary in the format of an interactive Web page (so independent of all material or software owner) in which he will be able to define, create and enhance his personal view.

Before starting the generation of the view, the user must specify his profile. If the views are associated with this profile, we will display their description to reuse the existing views and to avoid redundancy.



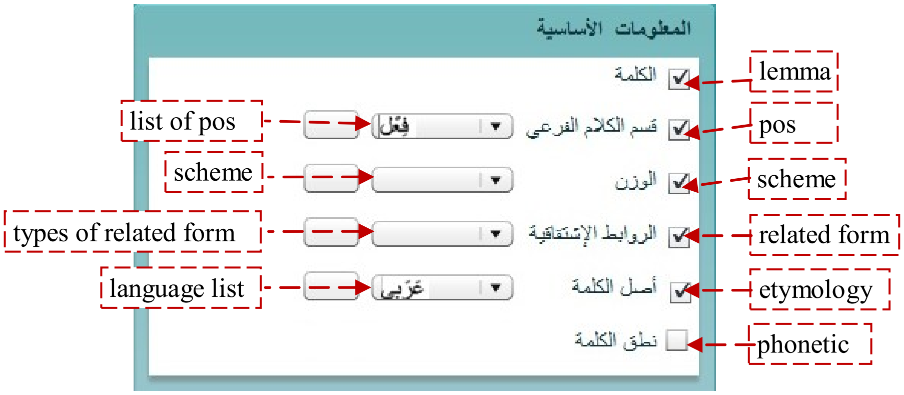
معلومات خاصة بالمستعمل

القسم غير محدد ▾ المستوى غير محدد ▾ المجال عام ▾

Figure 3. Specification of the user profile.

The user must set its category, its level and its field.

He can select an existing view corresponding to its needs or he can specify its purpose without using the existing. Then, he chooses the categories of data needed. Next, he can set their values.

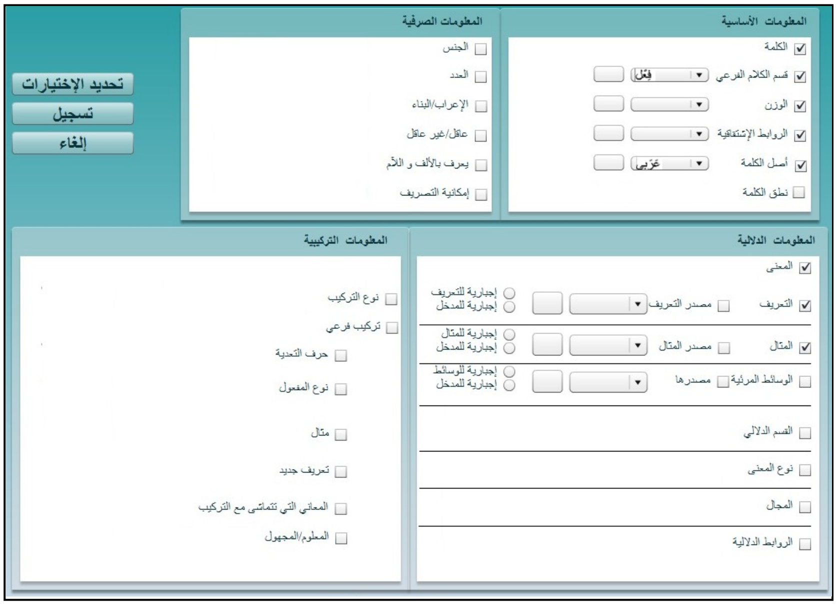


المعلومات الأساسية

☑ الكلمة
☑ قسم الكلام الفرعي فِعْل ▾
☑ الوزن ▾
☑ الروابط الإشتقاقية ▾
☑ أصل الكلمة عَرَبِي ▾
☐ نطق الكلمة

Figure 4. Representation of some categories of data and the list of values for selected DCs

In the following Figure 5, we present the key information in the dictionary. We give an example of view that includes only the schema, the derivational relations, sense, examples of all the Arabic verbs (pos = verb). Then from 38000 lexical entries, our view has 7000 verbs and 3000 roots i.e. only 10,000 entries.



المعلومات الأساسية

☑ الكلمة
☑ قسم الكلام الفرعي فِعْل ▾
☑ الوزن ▾
☑ الروابط الإشتقاقية ▾
☑ أصل الكلمة عَرَبِي ▾
☐ نطق الكلمة

المعلومات الصرفية

☐ الجنس
☐ العدد
☐ الإعراب/البناء
☐ عاقل/غير عاقل
☐ يعرف بالألف و اللاّم
☐ إمكانية التصريف

المعلومات الدلالية

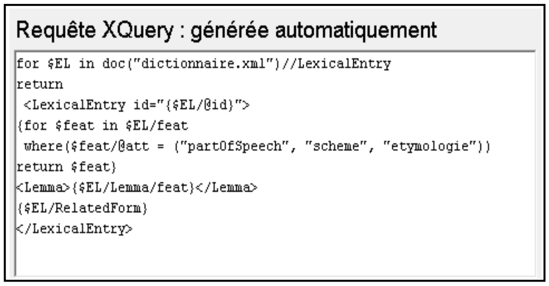
☑ المعنى
☑ التعريف ☐ مصدر التعريف ▾ ○ إجبارية للتعريف ○ إجبارية للمدخل
☑ المثال ☐ مصدر المثال ▾ ○ إجبارية للمثال ○ إجبارية للمدخل
☐ الوسائط المرئية ☐ مصدرها ▾ ○ إجبارية للوسائط ○ إجبارية للمدخل
☐ القسم الدلالي
☐ نوع المعنى
☐ المجال
☐ الروابط الدلالية

المعلومات التركيبية

☐ نوع التركيب
☐ تركيب فرعي
☐ حرف التعدية
☐ نوع المفعول
☐ مثال
☐ تعريف جديد
☐ المعاني التي تتماشى مع التركيب
☐ المعلوم/المجهول

Figure 5. Interface for creating a view.

In the Figure 5, the user has selected the lemma, pos, schema, the derivational relations, etymology, sense, definition and example. For pos, he fixed the value of “فِعْل” (verb). According to the specification of requirements, the user clicks the save button. The system checks the consistency of the chosen data categories and values, then, it generates a query in the XQuery language (see Figure 6).



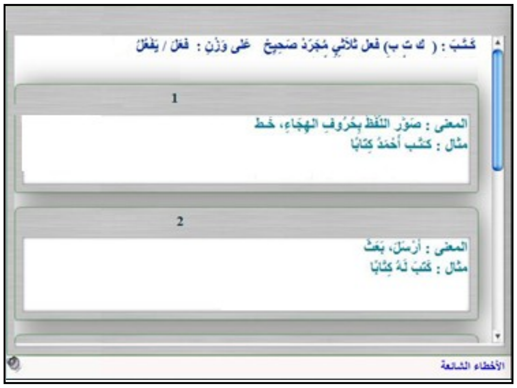
Requête XQuery : générée automatiquement

```
for $EL in doc("dictionnaire.xml")//LexicalEntry
return
 <LexicalEntry id="{$EL/@id}">
{for $feat in $EL/feat
 where($feat/@att = ("partOfSpeech", "scheme", "etymologie"))
return $feat}
<Lemma>{$EL/Lemma/feat}</Lemma>
{$EL/RelatedForm}
</LexicalEntry>
```

Figure 6. Example of a generated query

If the user chooses a materialized view, the system must save the query result in the user's computer after refinement (add missing information to validate the XML document).

For the verification of results, the user must choose the view before starting the search in the dictionary. In the following Figure 7, we present the results of research in the view already specified in Figure 5.



كَتَبَ : (ك ت ب) فعل ثلاثي مُجَرَّد صَحِيح   على وَزْن : فَعَلَ / يَفْعُلُ

1

المعنى : صَوَّرَ اللَّفظ بِحُرُوفِ الهِجَاءِ، خَطَّ
مثال : كَتَبَ أَحْمَدُ كِتَابًا

2

المعنى : أَرْسَلَ، بَعَثَ
مثال : كَتَبَ لَهُ كِتَابًا

الأخطاء الشائعة

Figure 7. Displayed results of query applied on a generated view.

6 Conclusion

The construction of specialized dictionaries is an old concept. However, it has not been used after the publication of LMF standard in spite of the complexity and the richness of normalized dictionaries. In this paper, we proposed an approach allowing the generation of specialized and personalized views of dictionaries according to users' profiles in order to benefit from the management of a unique dictionary and give appropriate services.

A Practical experiment was carried out on a normalized Arabic dictionary using an appropriate tool that permits to manage users' profiles and views' generation. We successfully performed some empirical illustrations starting from the normalized dictionary.

In the future, we will consider the experimentation of the developed tool in the generation of various personalized views both in virtual and materialized versions. Also, we plan to put up our system on the Web. Also, we plan to experiment our approach on others languages.

References

Baccar, F., Khemakhem, A., Gargouri, B., Haddar, K. and Hamadou, A.B.. 2008. *LMF Standardized Model for the Editorial Electronic Dictionaries of Arabic*. In Proceedings of NLPCS'08, pp. 64–73. Barcelona, Spain.

Ben Abderrahmen M., Gargouri B. and Jmaiel M. 2007. *LMF-QL: A Graphical Tool to Query LMF Databases for NLP and Editorial Use*. Book Chapter In: Human Language Technology. Challenges of the Information Society, Third Language and Technology Conference, LTC 2007, Poznan, Poland.

Corby O., Dieng-Kuntz R., Faron-Zucker C., Gandon F., Giboin A. 2005. *Le moteur de recherche sémantique Corese*. In Proc. of the Workshop Raisonner le web sémantique avec des graphes, AFIA platform. Nice.

Francopoulo G. and George M. 2008. *ISO/TC 37/SC 4 N453 (N330 Rev.16), Language resource management- Lexical markup framework (LMF)*.

Kemps-Snijders M., Nederhof M.-J. and Wittenburg P. 2006. *LEXUS, "A web-based tool for manipulating lexical resources"*. In LREC 2006.

Kirsch K. 2005. *LEXUS Manual*. LEXUS version 1.0.

LIRICS (Linguistic Infrastructure for Interoperable Resource and Systems). 2005. "*Guidelines and tools for producing standards, test-suites and API(s)*".

Véronis, J. and Ide, N.1996. *Encodage des dictionnaires électroniques: problèmes et propositions de la TEI*. In D. Piotrowsky (Ed.), Lexicograpbie et informatique - Autour de l'informatisation du Trésor de la Langue Française. Actes du Colloque International de Nancy (1996) (pp. 239-261). Paris: Didier Erudition.

Generating Confusion Sets for Context-Sensitive Error Correction

Alla Rozovskaya and Dan Roth
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{rozovska,danr}@illinois.edu

Abstract

In this paper, we consider the problem of generating candidate corrections for the task of correcting errors in text. We focus on the task of correcting errors in preposition usage made by non-native English speakers, using discriminative classifiers. The standard approach to the problem assumes that the set of candidate corrections for a preposition consists of all preposition choices participating in the task. We determine likely preposition confusions using an annotated corpus of non-native text and use this knowledge to produce smaller sets of candidates.

We propose several methods of restricting candidate sets. These methods exclude candidate prepositions that are not observed as valid corrections in the annotated corpus and take into account the likelihood of each preposition confusion in the non-native text. We find that restricting candidates to those that are observed in the non-native data improves both the precision and the recall compared to the approach that views all prepositions as possible candidates. Furthermore, the approach that takes into account the likelihood of each preposition confusion is shown to be the most effective.

1 Introduction

We address the problem of generating candidate corrections for the task of correcting context-dependent mistakes in text, mistakes that involve confusing valid words in a language. A well-studied instance of this problem – context-sensitive spelling errors – has received a lot of attention in natural language research (Golding and Roth, 1999; Carlson et al., 2001; Carlson and Fette, 2007; Banko and Brill, 2001). The context-sensitive spelling correction task addresses the problem of correcting spelling mistakes that result in legitimate words, such as confusing *their* and *there* or *your* and *you're*. In this task, a *candidate set* or a *confusion set* is defined that specifies a list of confusable words, e.g., {*their, there*} or {*cite, site, sight*}. Each occurrence of a confusable word in text is represented as a vector of features derived from a small context window around the target. A classifier is trained on text assumed to be error-free, replacing each target word occurrence (e.g. *their*) with a *confusion set* consisting of {*their, there*}, thus generating both positive and negative examples, respectively, from the same context. Given a text to correct, for each word in text that belongs to the confusion set the classifier predicts the most likely candidate in the confusion set.

More recently, work in error correction has taken an interesting turn and focused on correcting errors made by English as a Second Language (ESL) learners, with a special interest given to errors in article and preposition usage. These mistakes are some of the most common mistakes for non-native English speakers of all proficiency levels (Dalgish, 1985; Bitchener et al., 2005; Leacock et al., 2010). Approaches to correcting these mistakes have adopted the methods of the context-sensitive spelling correction task. A system is usually trained on well-formed native English text (Izumi et al., 2003; Eeg-Olofsson and Knuttson, 2003; Han et al., 2006; Felice and Pulman, 2008; Gamon et al., 2008; Tetreault

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 961–970,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

and Chodorow, 2008; Elghaari et al., 2010; Tetreault et al., 2010), but several works incorporate into training error-tagged data (Gamon, 2010; Han et al., 2010) or error statistics (Rozovskaya and Roth, 2010b). The classifier is then applied to non-native text to predict the correct article/preposition in context. The possible candidate selections include the set of all articles or all prepositions.

While in the article correction task the candidate set is small (a, *the*, no article), systems for correcting preposition errors, even when they consider the most common prepositions, may include between 9 to 34 preposition classes. For each preposition in the non-native text, every other candidate in the confusion set is viewed as a potential correction. This approach, however, does not take into account that writers do not make mistakes randomly: Not all candidates are equally likely given the preposition chosen by the author and errors may depend on the first language (L1) of the writer. In this paper, we define *L1-dependent candidate sets* for the preposition correction task (Section 4.1). L1-dependent candidate sets reflect preposition confusions observed with the speakers of the first language L1. We propose methods of enforcing L1-dependent candidate sets in training and testing.

We consider mistakes involving the top ten English prepositions. As our baseline system, we train a multi-class classifier in *one-vs-all* approach, which is a standard approach to multi-class classification. In this approach, a separate binary classifier for each preposition p_i, $1 \leq i \leq 10$, is trained, s.t. all p_i examples are positive examples for the classifier and all other nine classes act as negative examples. Thus, for each preposition p_i in non-native text there are ten[1] possible prepositions that the classifier can propose as corrections for p_i.

We contrast this baseline method to two methods that enforce L1-dependent candidate sets in training. First, we train a separate classifier for each preposition p_i on the prepositions that belong to *L1-dependent candidate set* of p_i. In this setting, the negative examples for p_i are those that belong to *L1-dependent candidate set* of p_i.

The second method of enforcing *L1-dependent candidate sets* in training is to train on native data with artificial preposition errors in the spirit of Rozovskaya and Roth (2010b), where the errors mimic the error rates and error patterns of the non-native text. This method requires more knowledge, since it uses a distribution of errors from an error-tagged corpus.

We also propose a method of enforcing *L1-dependent candidate sets* in testing, through the use of a confidence threshold. We consider two ways of applying a threshold: (1) the standard way, when a correction is proposed only if the classifier's confidence is sufficiently high and (2) L1-dependent threshold, when a correction is proposed only if it belongs to *L1-dependent candidate set*.

We show that the methods of restricting candidate sets to L1-dependent confusions improve the preposition correction system. We demonstrate that restricting candidate sets to those prepositions that are confusable in the data by L1 writers is beneficial, when compared to a system that assumes an unrestricted candidate set by considering as valid corrections all prepositions participating in the task. Furthermore, we find that the most effective method is the one that uses knowledge about the likelihoods of preposition confusions in the non-native text introduced through artificial errors in training.

The rest of the paper is organized as follows. First, we describe related work on error correction. Section 3 presents the ESL data and statistics on preposition errors. Section 4 describes the methods of restricting candidate sets in training and testing. Section 5 describes the experimental setup. We present and discuss the results in Section 6. The key findings are summarized in Table 5 and Fig. 1 in Section 6. We conclude with a brief discussion of directions for future work.

2 Related Work

Work in text correction has focused primarily on correcting context-sensitive spelling errors (Golding and Roth, 1999; Banko and Brill, 2001; Carlson et al., 2001; Carlson and Fette, 2007) and mistakes made by ESL learners, especially errors in article and preposition usage.

Roth (1998) takes a unified approach to resolving semantic and syntactic ambiguities in natural lan-

[1] This includes the preposition p_i itself. If proposed by the classifier, it would not be flagged as an error.

guage by treating several related problems, including word sense disambiguation, word selection, and context-sensitive spelling correction as instances of the disambiguation task. Given a *candidate set* or a *confusion set* of confusable words, the task is to select the most likely candidate in context. Examples of confusion sets are $\{$*sight, site, cite*$\}$ for context-sensitive spelling correction, $\{$*among, between*$\}$ for word selection, or a set of prepositions for the preposition correction problem.

Each occurrence of a candidate word in text is represented as a vector of features. A classifier is trained on a large corpus of error-free text. Given text to correct, for each word in text that belongs to the confusion set the classifier is used to predict the most likely candidate in the confusion set given the word's context.

In the same spirit, models for correcting ESL errors are generally trained on well-formed native text. Han et al. (2006) train a maximum entropy model to correct article mistakes. Chodorow et. al (2007), Tetreault and Chodorow (2008), and De Felice and Pulman (2008) train a maximum entropy model and De Felice and Pulman (2007) train a voted perceptron algorithm to correct preposition errors. Gamon et al. (2008) train a decision tree model and a language model to correct errors in article and preposition usage. Bergsma et al. (2009) propose a Naïve Bayes algorithm with web-scale N-grams as features, for preposition selection and context-sensitive spelling correction.

The set of valid candidate corrections for a target word includes all words in the confusion set. For the preposition correction task, the entire set of prepositions considered for the task is viewed as the set of possible corrections for each preposition in non-native text. Given a preposition with its surrounding context, the model selects the most likely preposition from the set of all candidates, where the set of candidates consists of nine (Felice and Pulman, 2008), 12 (Gamon, 2010), or 34 (Tetreault et al., 2010; Tetreault and Chodorow, 2008) prepositions.

2.1 Using Error-tagged Data in Training

Several recent works explore ways of using annotated non-native text when training error correction models.

One way to incorporate knowledge about which confusions are likely with ESL learners into the error correction system is to train a model on error-tagged data. Preposition confusions observed in the non-native text can then be included in training, by using the preposition chosen by the author (the *source* preposition) as a feature. This is not possible with a system trained on native data, because each *source* preposition is always the correct preposition.

Han et al. (2010) train a model on partially annotated Korean learner data. The error-tagged model trained on one million prepositions obtains a slightly higher recall and a significant improvement in precision (from 0.484 to 0.817) over a model fives times larger trained on well-formed text.

Gamon (2010) proposes a hybrid system for preposition and article correction, by incorporating the scores of a language model and class probabilities of a maximum entropy model, both trained on native data, into a meta-classifier that is trained on a smaller amount of annotated ESL data. The meta-classifier outperforms by a large margin both of the native models, but it requires large amounts of expensive annotated data, especially in order to correct preposition errors, where the problem complexity is much larger.

Rozovskaya and Roth (2010b) show that by introducing into native training data artificial article errors it is possible to improve the performance of the article correction system, when compared to a classifier trained on native data. In contrast to Gamon (2010) and Han et al. (2010) that use annotated data for training, the system is trained on native data, but the native data are transformed to be more like L1 data through artificial article errors that mimic the error rates and error patterns of non-native writers. This method is cheaper, since obtaining error statistics requires much less annotated data than training. Moreover, the training data size is not restricted by the amount of the error-tagged data available. Finally, the *source* article of the writer can be used in training as a feature, in the exact same way as with the models trained on error-tagged data, providing knowledge about which confusions are likely. Unlike article errors, preposition errors lend themselves very well to a study of confusion sets because the set of prepositions participating in the task is a lot bigger than the set of article choices.

3 ESL Data

3.1 Preposition Errors in Learner Data

Preposition errors are one of the most common mistakes that non-native speakers make. In the Cambridge Learner Corpus[2] (CLC), which contains data by learners of different first language backgrounds and different proficiency levels, preposition errors account for about 13.5% of all errors and occur on average in 10% of all sentences (Leacock et al., 2010). Similar error rates have been reported for other annotated ESL corpora, e.g. (Izumi et al., 2003; Rozovskaya and Roth, 2010a; Tetreault et al., 2010). Learning correct preposition usage in English is challenging for learners of all first language backgrounds (Dalgish, 1985; Bitchener et al., 2005; Gamon, 2010; Leacock et al., 2010).

3.2 The Annotated Corpus

We use data from an annotated corpus of essays written by ESL students. The essays were fully corrected and error-tagged by native English speakers. For each preposition used incorrectly by the author, the annotator also indicated the correct preposition choice. Rozovskaya and Roth (2010a) provide a detailed description of the annotation of the data.

The annotated data include sentences by speakers of five first language backgrounds: Chinese, Czech, Italian, Russian, and Spanish. The Czech, Italian, Russian and Spanish data come from the International Corpus of Learner English (ICLE, (Granger et al., 2002)), which is a collection of essays written by advanced learners of English. The Chinese data is a part of the Chinese Learners of English corpus (CLEC, (Gui and Yang, 2003)) that contains essays by students of all levels of proficiency. Table 1 shows preposition statistics based on the annotated data.

The combined data include 4185 prepositions, 8.4% of which were judged to be incorrect by the annotators. Table 1 demonstrates that the error rates in the Chinese speaker data, for which different proficiency levels are available, are 2 or 3 times higher than the error rates in other language groups. The data for other languages come from very advanced learners and, while there are also proficiency differences among advanced speakers, their error rates are much lower.

Source language	Total preps.	Incorrect preps.	Error rate
Chinese	953	144	15.1%
Czech	627	28	4.5%
Italian	687	43	6.3%
Russian	1210	85	7.0%
Spanish	708	52	7.3%
All	4185	352	8.4%

Table 1: **Statistics on prepositions in the ESL data.** Column *Incorrect* denotes the number of prepositions judged to be incorrect by the native annotators. Column *Error rate* denotes the proportion of prepositions used incorrectly.

We would also like to point out that we take as the *baseline*[3] for the task the accuracy of the non-native data, or the proportion of prepositions used correctly. Using the error rate numbers shown in Table 1, the baseline for Chinese speakers is thus 84.9%, and for all the data combined it is 91.6%.

3.3 Preposition Errors and L1

We focus on *preposition confusion* errors, mistakes that involve an incorrectly selected preposition[4]. We consider ten most frequent prepositions in English: *on, from, for, of, about, to, at, in, with,* and *by*[5].

We mentioned in Section 2 that not all preposition confusions are equally likely to occur and preposition errors may depend on the first language of the writer. Han et al. (2010) show that preposition errors in the annotated corpus by Korean learners are not evenly distributed, some confusions occurring more often than others. We also observe that confusion frequencies differ by L1. This is consistent with other studies, which show that learners' errors are influenced by their first language (Lee and Seneff, 2008; Leacock et al., 2010).

[2] http://www.cambridge.org/elt

[3] It is argued in Rozovskaya and Roth (2010b) that the most frequent class baselines are not relevant for error correction tasks. Instead, the error rate in the data need to be considered, when determining the baseline.

[4] We do not address errors of missing or extraneous prepositions.

[5] It is common to restrict the systems that detect errors in preposition usage to the top prepositions. In the CLC corpus, the usage of the ten most frequent prepositions accounts for 82% of all preposition errors (Leacock et al., 2010).

4 Methods of Improving Candidate Sets

In this section, we describe methods of restricting candidate sets according to the first language of the writer. For the preposition correction task, the standard approach considers all prepositions participating in the task as valid corrections for every preposition in the non-native data.

In Section 3.3, we pointed out that (1) not all preposition confusions are equally likely to occur and (2) preposition errors may depend on the first language of the writer. The methods of restricting confusion sets proposed in this work use knowledge about which prepositions are confusable based on the data by speakers of language L1.

We refer to the preposition originally chosen by the author in the non-native text as the *source* preposition, and *label* denotes the correct preposition choice, as chosen by the annotator. Consider, for example, the following sentences from the annotated corpus.

1. We ate **by*/with** our hands .
2. To tell the truth , time spent in jail often changes prisoners **to*/for** the worse.
3. And the problem that immediately appeared was that men were unable to cope **with** the new woman image .

In example 1, the annotator replaced *by* with *with*; *by* is the *source* preposition and *with* is the *label*. In example 2, *to* is the *source* and *for* is the *label*. In example 3, the preposition *with* is judged as correct. Thus, *with* is both the *source* and the *label*.

4.1 L1-Dependent Confusion Sets

Let source preposition p_i denote a preposition that appears in the data by speakers of L1. Let *ConfSet* denote the set of all prepositions that the system can propose as a correction for source preposition p_i. We define two types of confusion sets *ConfSet*. An unrestricted confusion set *AllConfSet* includes all ten prepositions. *L1-dependent confusion set L1ConfSet*(p_i) is defined as follows:

Definition $L1ConfSet(p_i) = \{p_j | \exists$ a sentence in which an L1 writer replaced preposition p_j with p_i $\}$

For example, in the Spanish speaker data, *from* is used incorrectly in place of *of* and *for*. Then for Spanish speakers, *L1ConfSet*(from)={from, of, for}.

Source prep. p_i	***L1ConfSet***(p_i)
on	{on, about, of, to, at, in, with, by}
by	{with, by, in}
from	{of, from, for}

Table 2: L1-dependent confusion sets for three prepositions based on data by Chinese speakers.

Table 2 shows for Chinese speakers three prepositions and their L1-dependent confusion sets.

We now describe methods of enforcing *L1-dependent confusion sets* in training and testing.

4.2 Enforcing L1-dependent Confusion Sets in Training

We propose two methods of enforcing L1-dependent confusion sets in training. They are contrasted to the typical method of training a multi-class 10-way classifier, where each class corresponds to one of the ten participating prepositions.

First, we describe the typical training setting.

NegAll Training proceeds in a standard way of training a multi-class classifier (*one-vs-all* approach) on all ten prepositions using well-formed native English data. For each preposition p_i, p_i examples are positive and the other nine prepositions are negative examples.

We now describe two methods of enforcing L1-dependent confusion sets in training.

NegL1 This method explores the difference between training with nine types as negative examples and (fewer than nine) L1-dependent negative examples.

For every preposition p_i, we train a classifier using only examples that are in *L1ConfSet*(p_i). In contrast to *NegAll*, for each source preposition, the negative examples are not all other nine types, but only those that belong in *L1ConfSet*(p_i). For each language L1, we train ten classifiers, one for each source preposition. For source preposition p_i in test, we consult the classifier for p_i. In this model, the confusion set for source p_i is restricted through training, since for source p_i, the possible candidate replacements are only those that the classifier sees in training, and they are all in *L1ConfSet*(p_i).

Training data	**Negative examples**	
	NegAll	**NegL1**
Clean	NegAll-Clean	NegL1-Clean
ErrorL1	NegAll-ErrorL1	-

Table 3: Training conditions that result in unrestricted (All) and L1-dependent training paradigms.

ErrorL1 This method restricts the candidate set to *L1ConfSet*(p_i) by generating artificial preposition errors in the spirit of Rozovskaya and Roth (2010b). The training data are thus no longer well-formed or *clean*, but augmented with L1 error statistics. Specifically, each preposition p_i in training is replaced with a different preposition p_j with probability *probConf*, s.t.

$$probConf = prob(p_i|p_j) \qquad (1)$$

Suppose 10% of all source prepositions *to* in the Russian speaker data correspond to label *for*. Then *for* is replaced with *to* with probability 0.1.

The classifier uses in training the source preposition as a feature, which cannot be done when training on well-formed text, as discussed in Section 2.1. By providing the source preposition as a feature, we enforce L1-dependent confusion sets in training, because the system learns which candidate corrections occur with source preposition p_i. An important distinction of this approach is that it does not simply provide L1-dependent confusion sets in training: Because errors are generated using L1 writers' error statistics, the likelihood of each candidate correction is also provided. This approach is also more knowledge-intensive, as it requires annotated data to obtain error statistics.

It should be noted that this method is orthogonal to the *NegAll* and *NegL1* methods of training described above and can be used in conjunction with each of them, only that it transforms the training data to account in a more natural way for ESL writing.

We combine the proposed methods *NegAll*, *NegL1* with the *Clean* or *ErrorL1* methods and create three training approaches shown in Table 3.

4.3 Restricting Confusion Sets in Testing

To reduce the number of false alarms, correction systems generally use a threshold on the confidence of the classifier, following (Carlson et al., 2001), and propose a correction only when the confidence of the classifier is above the threshold. We show in Section 5 that the system trained on data with artificial errors performs competitively even without a threshold. The other systems use a threshold. We consider two ways of applying a threshold[6]:

1. **ThreshAll** A correction for source preposition p_i is proposed only when the confidence of the classifier exceeds the threshold. For each preposition in the non-native data, this method considers *all* candidates as valid corrections.

2. **ThreshL1Conf** A correction for source preposition p_i is proposed only when the confidence of the classifier exceeds the empirically found threshold and the preposition proposed as a correction for p_i is in the confusion set *L1ConfSet*(p_i).

5 Experimental Setup

In this section, we describe experiments with L1-dependent confusion sets. Combining the three training conditions shown in Table 3 with the two ways of thresholding described in Section 4.3, we build four systems[7]:

1. **NegAll-Clean-ThreshAll** This system assumes both in training and in testing stages that all preposition confusions are possible. The system is trained as a multi-class 10-way classifier, where for each preposition p_i, all other nine prepositions are negative examples. In testing, when applying the threshold, all prepositions are considered as valid corrections.

2. **NegAll-Clean-ThreshL1** This system is trained exactly as *NegAll-Clean-ThreshAll* but in testing only corrections that belong

[6] Thresholds are found empirically: We divide the evaluation data into three equal parts and to each part apply the threshold, which is optimized on the other two parts of the data.

[7] In testing, it is not possible to consider a confusion set larger than the one used in training. Therefore, *ThreshAll* is only possible with *NegAll* training condition.

to *L1ConfSet*(p_i) are considered as valid corrections for p_i.

3. **NegL1-Clean-ThreshL1** For each preposition p_i, a separate classifier is trained on the prepositions that are in *L1ConfSet*(p_i), where p_i examples are positive and a set of (fewer than nine) p_i-dependent prepositions are negative. Only corrections that belong to *L1ConfSet*(p_i) are considered as valid corrections for p_i.[8] Ten p_i-dependent classifiers for each L1 are trained.

4. **NegAll-ErrorL1-NoThresh** A system is trained as a multi-class 10-way classifier with artificial preposition errors that mimic the errors rates and confusion patterns of the non-native text. For each L1, an L1-dependent system is trained. This system does not use a threshold. We discuss this in more detail below.

The system *NegAll-Clean-ThreshAll* is our *baseline* system. It assumes both in training and in testing that all preposition confusions are possible.

All of the systems are trained on the same set of word and part-of-speech features using the same set of training examples. Features are extracted from a window of eight words around the preposition and include words, part-of-speech tags and conjunctions of words and tags of lengths two, three, and four. Training data are extracted from English Wikipedia and the New York Times section of the Gigaword corpus (Linguistic Data Consortium, 2003).

In each training paradigm, we follow a discriminative approach, using an online learning paradigm and making use of the Averaged Perceptron Algorithm (Freund and Schapire, 1999) – we use the regularized version in Learning Based Java[9] (LBJ, (Rizzolo and Roth, 2007)). While classical Perceptron comes with generalization bound related to the margin of the data, Averaged Perceptron also comes with a PAC-like generalization bound (Freund and Schapire, 1999). This linear learning algorithm is known, both theoretically and experimentally, to be among the best linear learning approaches and is competitive with SVM and Logistic Regression, while being more efficient in training. It also has been shown to produce state-of-the-art results on many natural language applications (Punyakanok et al., 2008).

6 Results and Discussion

Table 4 shows performance of the four systems by the source language. For each source language, the methods that restrict candidate sets in training or testing outperform the baseline system *NegAll-Clean-ThreshAll* that does not restrict candidate sets. The *NegAll-ErrorL1-NoThresh* system performs better than the other three systems for all languages, except for Italian. In fact, for the Czech speaker data, all systems other than *NegAll-ErrorL1-NoThresh*, have a precision and a recall of 0, since no errors are detected[10].

Source lang.	System	Acc.	P	R
CH	*NegAll-Clean-ThreshAll*	84.78	47.58	11.46
	NegAll-Clean-ThreshL1	84.84	48.05	15.28
	NegL1-Clean-ThreshL1	84.94	50.87	11.46
	NegAll-ErrorL1-NoThresh	86.36	**55.27**	**27.43**
	`Baseline`	84.89		
CZ	*NegAll-Clean-ThreshAll*	94.74	0.00	0.00
	NegAll-Clean-ThreshL1	94.98	0.00	0.00
	NegL1-Clean-ThreshL1	94.66	0.00	0.00
	NegAll-ErrorL1-NoThresh	95.85	**75.00**	**10.71**
	`Baseline`	95.53		
IT	*NegAll-Clean-ThreshAll*	93.23	26.14	8.14
	NegAll-Clean-ThreshL1	94.03	**51.59**	**18.60**
	NegL1-Clean-ThreshL1	93.16	35.00	16.28
	NegAll-ErrorL1-NoThresh	93.60	44.95	10.47
	`Baseline`	93.74		
RU	*NegAll-Clean-ThreshAll*	92.73	31.11	3.53
	NegAll-Clean-ThreshL1	93.02	48.81	8.24
	NegL1-Clean-ThreshL1	92.44	34.42	8.82
	NegAll-ErrorL1-NoThresh	93.14	**52.38**	**12.94**
	`Baseline`	92.98		
SP	*NegAll-Clean-ThreshAll*	91.95	26.14	5.77
	NegAll-Clean-ThreshL1	92.02	28.64	5.77
	NegL1-Clean-ThreshL1	92.44	40.00	7.69
	NegAll-ErrorL1-NoThresh	93.71	**77.50**	**19.23**
	`Baseline`	92.66		

Table 4: **Performance results for the 4 systems**. All systems, except for *NegAll-ErrorL1-NoThresh*, use a threshold, which is optimized for accuracy on the development set. *Baseline* denotes the percentage of prepositions used correctly in the data. The baseline allows us to evaluate the systems with respect to accuracy, the percentage of prepositions, on which the prediction of the system is the same as the label. Averaged results over 2 runs.

[8] *ThreshAll* is not possible with this training option, as the system never proposes a correction that is not in *L1ConfSet*(p_i).

[9] LBJ code is available at `http://cogcomp.cs.illinois.edu/page/software`

[10] The Czech data set is the smallest and contains a total of 627 prepositions and only 28 errors.

The *NegAll-ErrorL1-NoThresh* system does not use a threshold. However, as shown in Fig. 1, it is possible to increase the precision of the *NegAll-ErrorL1-NoThresh* system by applying a threshold, at the expense of a lower recall.

While the ordering of the systems with respect to quality is not consistent from Table 4, due to modest test data sizes, Table 5 and Fig. 1 show results for the models on all data combined and thus give a better idea of how the systems compare against each other.

Table 5 shows performance results for all data combined. Both *NegAll-Clean-ThreshL1* and *NegL1-Clean-ThreshL1* achieve a better precision and recall over the system with an unrestricted candidate set *NegAll-Clean-ThreshAll*. Recall that both of the systems restrict candidate sets, the former at testing stage, the latter by training a separate classifier for each source preposition. *NegAll-Clean-ThreshL1* performs slightly better than *NegL1-Clean-ThreshL1*. We hypothesize that the *NegAll-Clean-ThreshAll* performance may be affected because the classifiers for different source prepositions contain different number of classes, depending on the size of *L1ConfSet* confusion sets, which makes it more difficult to find a unified threshold. The best performing system overall is *NegAll-ErrorL1-NoThresh*. While *NegAll-Clean-ThreshL1* and *NegL1-Clean-ThreshL1* restrict candidate sets, *NegAll-ErrorL1-NoThresh* also provides information about the likelihood of each confusion, which benefits the classifier. The differences between *NegAll-ErrorL1-ThreshL1* and each of the other three systems are statistically significant[11] (McNemar's test, $p < 0.01$). The table also demonstrates that the results on the correction task may vary widely. For example, the recall varies by language between 10.47% and 27.43% for the *NegAll-ErrorL1-NoThresh system*. The highest recall numbers are obtained for Chinese speakers. These speakers also have the highest error rate, as we noted in Section 3.

System	Acc.	P	R
NegAll-Clean-ThreshAll	90.90	31.11	7.95
NegAll-Clean-ThreshL1	91.11	37.82	12.78
NegL1-Clean-ThreshL1	90.97	34.34	9.66
NegAll-ErrorL1-NoThresh	92.23	**58.47**	**19.60**

Table 5: **Comparison of the performance of the 4 systems on all data combined**. All systems, except for *NegAll-ErrorL1-NoThresh*, use a threshold, which is optimized for accuracy on the development set. The differences between *NegAll-ErrorL1-ThreshL1* and each of the other three systems are statistically significant (McNemar's test, $p < 0.01$).

Finally, Fig. 1 shows precision/recall curves for the systems[12]. The curves are obtained by varying a decision threshold for each system. Before we examine the differences between the models, it should be noted that in error correction tasks precision is favored over recall due to the low level of error.

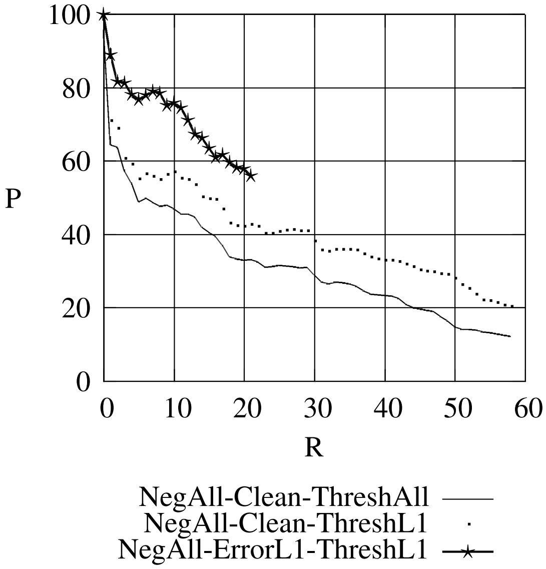


Figure 1: Precision and recall (%) for three models: *NegAll-Clean-ThreshAll*, *NegAll-Clean-ThreshL1*, and *NegAll-ErrorL1-ThreshL1*.

The curves demonstrate that *NegAll-Clean-ThreshL1* and *NegAll-ErrorL1-ThreshL1* are superior to the baseline system *NegAll-Clean-ThreshAll*: on the same recall points, the precision for both systems is consistently better than for the base-

[11] Tests of statistical significance compare the combined results from all language groups for each model. For example, to compare the model *NegAll-Clean-ThreshAll* to *NegAll-ErrorL1-NoThresh*, we combine the results from the five language-specific models *NegAll-ErrorL1-NoThresh* and compare them to the results on the combined data from the five language groups achieved by the model *NegAll-Clean-ThreshAll*.

[12] *NegL1-Clean-ThreshL1* is not shown, since it is similar in its behavior to *NegAll-Clean-ThreshL1*.

line model[13]. Moreover, while restricting candidate sets improves the results, providing information to the classifier about the likelihoods of different confusions is more helpful, which is reflected in the precision differences between *NegAll-Clean-ThreshL1* and *NegAll-ErrorL1-ThreshL1*. In fact, *NegAll-ErrorL1-ThreshL1* achieves a higher precision compared to the other systems, even when no threshold is used (Tables 4 and 5). This is because, unlike the other models, this system does not tend to propose too many false alarms.

6.1 Comparison to Other Systems

It is difficult to compare performance to other systems, since training and evaluation are not performed on the same data, and results may vary widely depending on the first language and proficiency level of the writer. However, in Table 6 we list several systems and their performance on the task. Tetreault et al. (2010) train on native data and obtain a precision of 48.6% and a recall of 22.5% with top 34 prepositions on essays from the Test of English as a Foreign Language exams. Han et al. (2010) obtain a precision of 81.7% and a recall of 13.2% using a model trained on partially error-tagged data by Korean speakers on top ten prepositions. A model trained on 2 million examples from clean text achieved on the same data set a precision of 46.3% and a recall of 11.6%.

Gamon (2010) shows precision/recall curves on the combined task of detecting missing, extraneous and confused prepositions. For recall points 10% and 20%, precisions of 55% and 40%, respectively, are obtained. For our data, a recall of 10% corresponds to a precision of 46% for the worst-performing model and 78% for the best-performing model. For 20% recall, we obtain a precision of 33% for the worst-performing model and 58% for the best-performing model. We would like to emphasize that these comparisons should be interpreted with caution.

7 Conclusion and Future Work

In this paper, we proposed methods for improving candidate sets for the task of detecting and correcting errors in text. To correct errors in preposition usage made by non-native speakers of English, we proposed L1-dependent confusion sets that determine valid candidate corrections using knowledge about preposition confusions observed in the non-native text. We found that restricting candidates to those that are observed in the non-native data improves both the precision and the recall compared to a classifier that considers as possible candidates the set of all prepositions. Furthermore, the approach that takes into account the likelihood of each preposition confusion is shown to be the most effective.

System	**Training Data**	**P**	**R**
Tetreault et al., 2010	native; 34 preps.	48.6	22.5
Han et al., 2010	partially error-tagged; 10 preps.	81.7	13.2
Han et al., 2010	native; 10 preps.	46.3	11.6
Gamon, 2010	native; 12 preps.+ extraneous+missing	33.0	10.0
Gamon, 2010	native+error-tagged; 12 preps.+ extraneous+missing	55.0	10.0
NegAll-Clean-ThreshAll	native; 10 preps.	46.0	10.0
NegAll-ErrorL1-ThreshL1	native with L1 error statistics; 10 preps.	78.0	10.0

Table 6: **Comparison to other systems**. Please note that a direct comparison is not possible, since the systems are trained and evaluated on different data sets. Gamon (2010) also considers missing and extraneous preposition errors.

The methods proposed in this paper make use of select characteristics that the error-tagged data can provide. We would also like to compare the proposed methods to the quality of a model trained on error-tagged data. Improving the system is also in our future work, but orthogonal to the current contribution.

Acknowledgments

We thank Nick Rizzolo for helpful discussions on LBJ. We also thank Peter Chew and the anonymous reviewers for their insightful comments. This research is partly supported by a grant from the U.S. Department of Education.

[13] While significance tests did not show differences between *NegAll-Clean-ThreshAll* and *NegAll-Clean-ThreshL1*, perhaps due to a modest test set size, the curves demonstrate that the latter system indeed provides a stable advantage over the baseline unrestricted approach.

References

M. Banko and E. Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 26–33, Toulouse, France, July.

J. Bitchener, S. Young, and D. Cameron. 2005. The effect of different types of corrective feedback on ESL student writing. *Journal of Second Language Writing*.

A. Carlson and I. Fette. 2007. Memory-based context-sensitive spelling correction at web scale. In *Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA)*.

A. J. Carlson, J. Rosen, and D. Roth. 2001. Scaling up context sensitive text correction. In *Proceedings of the National Conference on Innovative Applications of Artificial Intelligence (IAAI)*, pages 45–50.

M. Chodorow, J. Tetreault, and N.-R. Han. 2007. Detection of grammatical errors involving prepositions. In *Proceedings of the Fourth ACL-SIGSEM Workshop on Prepositions*, pages 25–30, Prague, Czech Republic, June. Association for Computational Linguistics.

G. Dalgish. 1985. Computer-assisted ESL research. *CALICO Journal*, 2(2).

J. Eeg-Olofsson and O. Knuttson. 2003. Automatic grammar checking for second language learners - the use of prepositions. *Nodalida*.

A. Elghaari, D. Meurers, and H. Wunsch. 2010. Exploring the data-driven prediction of prepositions in english. In *Proceedings of COLING 2010*, Beijing, China.

R. De Felice and S. Pulman. 2007. Automatically acquiring models of preposition use. In *Proceedings of the Fourth ACL-SIGSEM Workshop on Prepositions*, pages 45–50, Prague, Czech Republic, June.

R. De Felice and S. Pulman. 2008. A classifier-based approach to preposition and determiner error correction in L2 English. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 169–176, Manchester, UK, August.

Y. Freund and R. E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

M. Gamon, J. Gao, C. Brockett, A. Klementiev, W. Dolan, D. Belenko, and L. Vanderwende. 2008. Using contextual speller techniques and language modeling for ESL error correction. In *Proceedings of IJCNLP*.

M. Gamon. 2010. Using mostly native data to correct errors in learners' writing. In *NAACL*, pages 163–171, Los Angeles, California, June.

A. R. Golding and D. Roth. 1999. A Winnow based approach to context-sensitive spelling correction. *Machine Learning*, 34(1-3):107–130.

S. Granger, E. Dagneaux, and F. Meunier. 2002. *International Corpus of Learner English*. Presses universitaires de Louvain.

S. Gui and H. Yang. 2003. *Zhongguo Xuexizhe Yingyu Yuliaohu. (Chinese Learner English Corpus)*. Shanghai Waiyu Jiaoyu Chubanshe. (In Chinese).

N. Han, M. Chodorow, and C. Leacock. 2006. Detecting errors in English article usage by non-native speakers. *Journal of Natural Language Engineering*, 12(2):115–129.

N. Han, J. Tetreault, S. Lee, and J. Ha. 2010. Using an error-annotated learner corpus to develop and ESL/EFL error correction system. In *LREC*, Malta, May.

E. Izumi, K. Uchimoto, T. Saiga, T. Supnithi, and H. Isahara. 2003. Automatic error detection in the Japanese learners' English spoken data. In *The Companion Volume to the Proceedings of 41st Annual Meeting of the Association for Computational Linguistics*, pages 145–148, Sapporo, Japan, July.

C. Leacock, M. Chodorow, M. Gamon, and J. Tetreault. 2010. Morgan and Claypool Publishers.

J. Lee and S. Seneff. 2008. An analysis of grammatical errors in non-native speech in English. In *Proceedings of the 2008 Spoken Language Technology Workshop*.

V. Punyakanok, D. Roth, and W. Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2).

N. Rizzolo and D. Roth. 2007. Modeling Discriminative Global Inference. In *Proceedings of the First International Conference on Semantic Computing (ICSC)*, pages 597–604, Irvine, California, September. IEEE.

D. Roth. 1998. Learning to resolve natural language ambiguities: A unified approach. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 806–813.

A. Rozovskaya and D. Roth. 2010a. Annotating ESL errors: Challenges and rewards. In *Proceedings of the NAACL Workshop on Innovative Use of NLP for Building Educational Applications*.

A. Rozovskaya and D. Roth. 2010b. Training paradigms for correcting errors in grammar and usage. In *Proceedings of the NAACL-HLT*.

J. Tetreault and M. Chodorow. 2008. The ups and downs of preposition error detection in ESL writing. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 865–872, Manchester, UK, August.

J. Tetreault, J. Foster, and M. Chodorow. 2010. Using parse features for preposition selection and error detection. In *ACL*.

Confidence in Structured-Prediction using Confidence-Weighted Models

Avihai Mejer
Department of Computer Science
Technion-Israel Institute of Technology
Haifa 32000, Israel
amejer@tx.technion.ac.il

Koby Crammer
Department of Electrical Engineering
Technion-Israel Institute of Technology
Haifa 32000, Israel
koby@ee.technion.ac.il

Abstract

Confidence-Weighted linear classifiers (CW) and its successors were shown to perform well on binary and multiclass NLP problems. In this paper we extend the CW approach for sequence learning and show that it achieves state-of-the-art performance on four noun phrase chucking and named entity recognition tasks. We then derive few algorithmic approaches to estimate the prediction's correctness of each label in the output sequence. We show that our approach provides a reliable relative correctness information as it outperforms other alternatives in ranking label-predictions according to their error. We also show empirically that our methods output close to absolute estimation of error. Finally, we show how to use this information to improve active learning.

1 Introduction

In the past decade structured classification has seen much interest by the machine learning community. After the introduction of conditional random fields (CRFs) (Lafferty et al., 2001), and maximum margin Markov networks (Taskar et al., 2003), which are batch algorithms, new online method were introduced. For example the passive-aggressive algorithm was adapted to chunking (Shimizu and Haas, 2006), parsing (McDonald et al., 2005b), learning preferences (Wick et al., 2009) and text segmentation (McDonald et al., 2005a). These new online algorithms are fast to train and simple to implement, yet they generate models that output merely a prediction with no additional information, as opposed to probabilistic models like CRFs or HMMs.

In this work we fill this gap proposing few alternatives to compute confidence in the output of discriminative non-probabilistic algorithms. As before, our algorithms output the highest-scoring labeling. However, they also compute additional labelings, that are used to compute the *per word* confidence in its labelings. We build on the recently introduced confidence-weighted learning (Dredze et al., 2008; Crammer et al., 2009b) and induce a distribution over labelings from the distribution maintained over weight-vectors.

We show how to compute confidence estimates in the label predicted per word, such that the confidence reflects the probability that the label is not correct. We then use this confidence information to rank all labeled words (in all sentences). This can be thought of as a retrieval of the erroneous words, which can than be passed to human annotator for an examination, either to correct these mistakes or as a quality control component. Next, we show how to apply our techniques to active learning over sequences. We evaluate our methods on four NP chunking and NER datasets and demonstrate the usefulness of our methods. Finally, we report the performance of obtained by CW-like adapted to sequence prediction, which are comparable with current state-of-the-art algorithms.

2 Confidence-Weighted Learning

Consider the following online binary classification problem that proceeds in rounds. On the ith round the online algorithm receives an input $\boldsymbol{x}_i \in \mathbb{R}^d$ and

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 971–981,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

applies its current rule to make a prediction $\hat{y}_i \in \mathcal{Y}$, for the binary set $\mathcal{Y} = \{-1, +1\}$. It then receives the correct label $y_i \in \mathcal{Y}$ and suffers a loss $\ell(y_i, \hat{y}_i)$. At this point, the algorithm updates its prediction rule with the pair $(\boldsymbol{x}_i, y_i)$ and proceeds to the next round. A summary of online algorithms can be found in (Cesa-Bianchi and Lugosi, 2006).

Online confidence-weighted (CW) learning (Dredze et al., 2008; Crammer et al., 2008), generalized the passive-aggressive (PA) update principle to multivariate Gaussian distributions over the weight vectors - $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ - for binary classification. The mean $\boldsymbol{\mu} \in \mathbb{R}^d$ contains the current estimate for the best weight vector, whereas the Gaussian covariance matrix $\Sigma \in \mathbb{R}^{d \times d}$ captures the confidence in this estimate. More precisely, the diagonal elements $\Sigma_{p,p,}$ capture the confidence in the value of the corresponding weight $\boldsymbol{\mu}_p$; the smaller the value of $\Sigma_{p,p,}$ is, the more confident is the model in the value of $\boldsymbol{\mu}_p$. The off-diagonal elements $\Sigma_{p,q}$ for $p \neq q$ capture the correlation between the values of $\boldsymbol{\mu}_p$ and $\boldsymbol{\mu}_q$. When the data is of large dimension, such as in natural language processing, a model that maintains a full covariance matrix is not feasible and we back-off to diagonal covariance matrices.

CW classifiers are trained according to a PA rule that is modified to track differences in Gaussian distributions. At each round, the new mean and covariance of the weight vector distribution is chosen to be the solucion of an optimization problem (see (Crammer et al., 2008) for details). This particular CW rule may over-fit by construction. A more recent alternative scheme called AROW (adaptive regularization of weight-vectors) (Crammer et al., 2009b) replaces the guaranteed prediction at each round with the a more relaxed objective (see (Crammer et al., 2009b)). AROW has been shown to perform well in practice, especially for noisy data where CW severely overfits.

The solution for the updates of CW and AROW share the same general form,

$$\boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_i + \alpha_i \Sigma_i y_i \boldsymbol{x}_i \; ; \; \Sigma_{i+1}^{-1} = \Sigma_{i+1}^{-1} + \beta_i \boldsymbol{x}_i \boldsymbol{x}_i^\top \; , \quad (1)$$

where the difference between CW and AROW is the specific instance-dependent rule used to set the values of α_i and β_i.

Algorithm 1 Sequence Labeling CW/AROW

```
Input: Joint feature mapping $\mathbf{\Phi}(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^d$
       Initial variance $a > 0$
       Tradeoff Parameter $r > 0$ (AROW)
    or Confidence parameter $\phi$ (CW)
Initialize: $\boldsymbol{\mu}_0 = \mathbf{0}$ , $\Sigma_0 = aI$
for $i = 1, 2 \ldots$ do
    Get $\boldsymbol{x}_i \in \mathcal{X}$
    Predict best labeling
        $\hat{\boldsymbol{y}}_i = \arg\max_{\boldsymbol{z}} \boldsymbol{\mu}_{i-1} \cdot \mathbf{\Phi}(\boldsymbol{x}_i, \boldsymbol{z})$
    Get correct labeling $\boldsymbol{y}_i \in \mathcal{Y}^{|\boldsymbol{x}_i|}$
    Define $\mathbf{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}} = \mathbf{\Phi}(\boldsymbol{x}, \boldsymbol{y}_i) - \mathbf{\Phi}(\boldsymbol{x}, \hat{\boldsymbol{y}}_i)$
    Compute $\alpha_i$ and $\beta_i$ (Eq. (3) for CW ; Eqs. (4),$\beta_i = 1/r$) for AROW)
    Set $\boldsymbol{\mu}_i = \boldsymbol{\mu}_{i-1} + \alpha_i \Sigma_{i-1} \mathbf{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}}$
    Set $\Sigma_i^{-1} = \Sigma_{i-1}^{-1} + \beta_i \mathbf{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}} \mathbf{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}}^\top$
end for
```

3 Sequence Labeling

In the sequence labeling setting, instances $\boldsymbol{x}$ belong to a general input space $\mathcal{X}$ and conceptually are composed of a finite number n of components, such as words of a sentence. The number of components $n = |\boldsymbol{x}|$ varies between instances. Each part of an instance is labelled from a finite set $\mathcal{Y}$, $|\mathcal{Y}| = K$. That is, a labeling of an entire instance belongs to the product set $\boldsymbol{y} \in \mathcal{Y} \times \mathcal{Y} \ldots \mathcal{Y}$ (n times).

We employ a general approach (Collins, 2002; Crammer et al., 2009a) to generalize binary classification and use a joined feature mapping of an instance $\boldsymbol{x}$ and a labeling $\boldsymbol{y}$ into a common vector space, $\mathbf{\Phi}(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^d$.

Given an input instance $\boldsymbol{x}$ and a model $\boldsymbol{\mu} \in \mathbb{R}^d$ we predict the labeling with the highest score, $\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{z}} \boldsymbol{\mu} \cdot \mathbf{\Phi}(\boldsymbol{x}, \boldsymbol{z})$. A brute-force approach evaluates the value of the score $\boldsymbol{\mu} \cdot \mathbf{\Phi}(\boldsymbol{x}, \boldsymbol{z})$ for each possible labeling $\boldsymbol{z} \in \mathcal{Y}^n$, which is not feasible for large values of n. Instead, we follow standard factorization and restrict the joint mapping to be of the form, $\mathbf{\Phi}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{p=1}^{n} \mathbf{\Phi}(\boldsymbol{x}, y_p) + \sum_{q=2}^{n} \mathbf{\Phi}(\boldsymbol{x}, y_q, y_{q-1})$. That is, the mapping is a sum of mappings, each taking into consideration only a label of a single part, or two consecutive parts. The time required to compute the $\max$ operator is linear in n and quadratic in K using the dynamic-programming Viterbi algorithm.

After the algorithm made a prediction, it uses the current labeled instance $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ to update the model. We now define the update rule both for a version of CW and for AROW for strucutred learning, staring with CW. Given the input parameter ϕ of CW we denote by $\phi' = 1 + \phi^2/2, \phi'' = 1 + \phi^2$. We follow a similar argument as in the single update of (Crammer et al., 2009a, sec. 5.1) to sequence labeling by a reduction to binary classification. We first define the difference between the feature vector associated with the current labeling $\boldsymbol{y}_i$ and the feature vector associated with some labeling $\boldsymbol{z}$ to be, $\boldsymbol{\Delta}_{i,\boldsymbol{y},\boldsymbol{z}} = \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}_i) - \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{z})$, and in particular, when we use the prediction $\hat{\boldsymbol{y}}_i$ we get, $\boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}} = \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}_i) - \boldsymbol{\Phi}(\boldsymbol{x}, \hat{\boldsymbol{y}}_i)$. The CW update is,

$$\begin{aligned} \boldsymbol{\mu}_i &= \boldsymbol{\mu}_{i-1} + \alpha_i \Sigma_{i-1} \boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}} \\ \Sigma_i^{-1} &= \Sigma_{i-1}^{-1} + \beta_i \boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}} \boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}}^\top , \end{aligned} \tag{2}$$

where the two scalars α_i and β_i are set using the update rule defined by (Crammer et al., 2008) for binary classification,

$$v_i = \boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}}^\top \Sigma_{i-1} \boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}} \ , \ m_i = \boldsymbol{\mu}_{i-1} \cdot \boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}} \tag{3}$$

$$\alpha_i = \max\left\{0, \frac{1}{v_i \phi''} \left(-m_i \phi' + \sqrt{m_i^2 \frac{\phi^4}{4} + v_i \phi^2 \phi''} \right) \right\}$$

$$\beta_i = \frac{\alpha_i \phi}{\sqrt{v_i^+}} \quad , \quad v_i^+ = \frac{1}{4} \left(-\alpha_i v_i \phi + \sqrt{\alpha_i^2 v_i^2 \phi^2 + 4 v_i} \right)^2$$

We turn our attention and describe a modification of AROW for sequence prediction. Replacing the binary-hinge loss in (Crammer et al., 2009b, Eqs. (1,2)) the first one with the corresponding multi-class hinge loss for structured problems we obtain, $\frac{1}{2} (\boldsymbol{\mu}_i - \boldsymbol{\mu})^\top \Sigma_i^{-1} (\boldsymbol{\mu}_i - \boldsymbol{\mu}) + \frac{1}{2r} (\max \{0, \max_{\boldsymbol{z} \neq \boldsymbol{y}} \{d(\boldsymbol{y}, \boldsymbol{z}) - \boldsymbol{\mu} \cdot (\boldsymbol{\Delta}_{i,\boldsymbol{y},\boldsymbol{z}})\}\})^2$, where, $d(\boldsymbol{y}, \boldsymbol{z}) = \sum_{q=1}^{|\boldsymbol{x}|} 1_{y_q \neq z_q}$, is the hamming distance between the two label sequences $\boldsymbol{y}$ and $\boldsymbol{z}$. The last equation is hard to optimize since the max operator is enumerating over exponential number of possible labellings $\boldsymbol{z}$. We thus approximate the enumeration over all possible $\boldsymbol{z}$ with the predicted label sequence $\hat{y}_i$ and get, $\frac{1}{2} (\boldsymbol{\mu}_i - \boldsymbol{\mu})^\top \Sigma_i^{-1} (\boldsymbol{\mu}_i - \boldsymbol{\mu}) + \frac{1}{2r} \left(\max \left\{ 0, d(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i) - \boldsymbol{\mu} \cdot \left(\boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}} \right) \right\} \right)^2$. Computing the optimal value of the last equation we get an update of the form of the first equation of Eq. (2) where

$$\alpha_i = \frac{\max \left\{ 0, d(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i) - \boldsymbol{\mu}_{i-1} \cdot \left(\boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}} \right) \right\}}{r + \boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}}^\top \Sigma_{i-1} \boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}}} . \tag{4}$$

Dataset	Sentences	Words	Features
NP chunking	11K	259K	1.35M
NER English	17.5K	250K	1.76M
NER Spanish	10.2K	317.6K	1.85M
NER Dutch	21K	271.5K	1.76M

Table 1: Properties of datasets.

	AROW	CW	5-best PA	Perceptron
NP chunking	0.946	0.947	0.946	**0.944
NER English	0.878	0.877	* 0.870	* 0.862
NER Dutch	0.791	0.787	0.784	* 0.761
NER Spanish	0.775	0.774	0.773	* 0.756

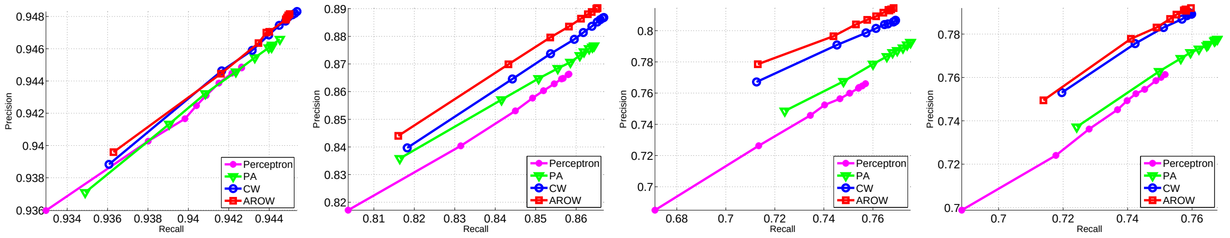
Table 2: Averaged F-measure of methods. Statistical significance (t-test) are with respect to AROW, where * indicates 0.001 and ** indicates 0.01

We proceed with the confidence paramters in (Crammer et al., 2009b, Eqs. (1,2)), which takes into considiration the change of confidence due to the update. The effective features vector that is used to update the mean parameters is $\boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}}$, and thus the structured update is, $\frac{1}{2} \log \left(\frac{\det \Sigma_i}{\det \Sigma} \right) + \frac{1}{2} \mathrm{Tr} \left(\Sigma_{i-1}^{-1} \Sigma \right) + \frac{1}{2r} \boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}}^\top \Sigma \boldsymbol{\Delta}_{i,\boldsymbol{y},\hat{\boldsymbol{y}}}$. Solving the above equation we get an update of the form of the second term of Eq. (2) where $\beta_i = \frac{1}{r}$. The pseudo-code of CW and AROW for sequence problems appears in Alg. 1.

4 Evaluation

For the experiments described in this paper we used four large sequential classification datasets taken from the CoNLL-2000, 2002 and 2003 shared tasks: noun-phrase (NP) chunking (Kim et al., 2000), and named-entity recognition (NER) in Spanish, Dutch (Tjong and Sang, 2002) and English (Tjong et al., 2003). The properties of the four datasets are summarized in Table 1. We followed the feature generation process of (Sha and Pereira, 2003).

Although our primary goal is estimating confidence in prediction and not the actual performance itself, we first report the results of using AROW and CW for sequence learning. We compared the performance CW and AROW of Alg. 1 with two standard online baseline algorithms: Averaged-Perceptron algorithm and 5-best PA (the value of five was shown to be optimal for various tasks (Crammer et al., 2005)). The update rule described in Alg. 1 assumes a full covariance matrix, which is not feasible in our



(a) NP Chunking (b) NER English (c) NER Dutch (d) NER Spanish

Figure 1: Precision and Recall on four datasets (four panels). Each connected set of ten points corresponds to the performance of a specific algorithm after each of the 10 iterations, increasing from bottom-left to top-right.

		Prec	Recall	F-meas	% Err
NP chunking	CW	0.945	0.942	0.943	2.34%
	CRF	0.938	0.934	0.936	2.66%
NER English	CW	0.838	0.826	0.832	3.38%
	CRF	0.823	0.820	0.822	3.53%
NER Dutch	CW	0.803	0.755	0.778	2.05%
	CRF	0.775	0.753	0.764	2.09%
NER Spanish	CW	0.738	0.720	0.729	4.09%
	CRF	0.751	0.730	0.740	2.05%

Table 3: Precision, Recall, F-measure and percentage of mislabeled words results of CW vs. CRF

setting. Three options are possible: compute a full Σ and then take its diagonal elements; compute a full inverse Σ, take its diagonal elements and then compute its inverse; assume that Σ is diagonal and compute the optimal update for this choice. We found the first method to work best, and thus employ it from now on.

The hyper parameters (r for AROW, ϕ for CW, C for PA) were tuned for each task by a single run over a random split of the data into a three-fourths training set and a one-fourth test set. We used parameter averaging with all methods.

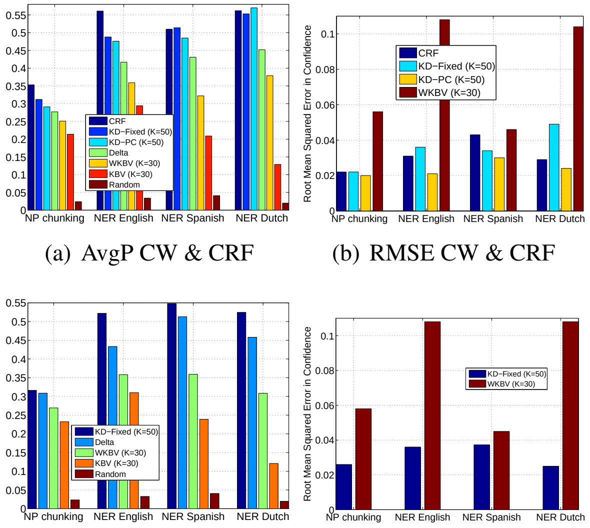
For each of the four datasets we used 10-fold cross validation. All algorithms (Perceptron, PA, CW and AROW) are online, and as mentioned above work in rounds. For each of the ten folds, each of the four algorithm performed ten (10) iterations over the training set and the performance (Recall, Precision and F-measure) was evaluated on the test set after each iteration.

The F-measure of the four algorithms after 10 iterations over the four datasets is summarized in Table 2. The general trend is that AROW slightly outperforms CW, which is better than PA that is better than the Perceptron. The difference between AROW and the Perceptron is significant, and between AROW and PA is significant in two datasets. The difference between AROW and CW is not significant although it is consistent.

We further investigate the convergence properties of the algorithms in Fig. 1. The figure shows the recall and precision results after each training round averaged across the 10 folds. Each panel summarizes the results on a single dataset, and in each panel a single set of connected points corresponds to one algorithm. Points in the left-bottom of the plot correspond to early iterations and points in the right-top correspond to later iterations. Long segments indicate a big improvement in performance between two consecutive iterations.

Few points are in order. First, high (in the y-axis) values indicate better precision and right (in the x-axis) values indicate better recall. Second, the performance of all algorithms is converging in about 10 iterations as indicated by the fact the points in the top-right of the plot are close to each other. Third, the long segments in the bottom-left for the Perceptron algorithm indicate that this algorithm benefits more from more than one pass compared with the other. Fourth, on the three NER datasets after 10 iterations AROW gets slightly higher precision values than CW, while CW gets slightly higher recall values than AROW. This is indicated by the fact that the top-right red square is left and above to the top-right blue circle. Finally, in two datasets, PA get slightly better recall than CW and AROW, but paying in terms of precision and overall F-measure performance.

In addition to online algorithms we also compared the performance of CW with the CRF algo-



(a) AvgP CW & CRF (b) RMSE CW & CRF

(c) AvgP PA (d) RMSE PA

Figure 2: Two left panels: average precision of rankings of the words of the test-set according to confidence in the prediction of seven methods (left to right bars in each group): CRF, KD-Fixed, KD-PC, Delta, WKBV, KBV and random ordering, when training with the CW algorithm (top) and the PA algorithm (bottom). Two right panels: The root-mean-squared-error of four methods that output absolute valued confidence: CRF, KD-Fixed, KD-PC and WKBV.

rithm which is a batch algorithm. We used Mallet toolkit (McCallum, 2002) for CRF implementation. For feature generation we used a combination of standard methods provided with Mallet toolkit (called pipes). We chose a combination yielding a feature set that is close as possible to the feature set we used in our system but it was not a perfect match, CRF generated about 20% fewer features in all datasets. Nevertheless, any other combination of pipes we tried only hurt CRF performance. The precision, recall, F-measure and percentage of mislabeled words of CW algorithm compared with CRF measured over a single split of the data into a three-fourths training set and a one-fourth test set is summarized in Table 3. We see that in three of the four datasets CW outperforms CRF and in one dataset CRF performs better. Some of the performance differences may be due to the differences in features.

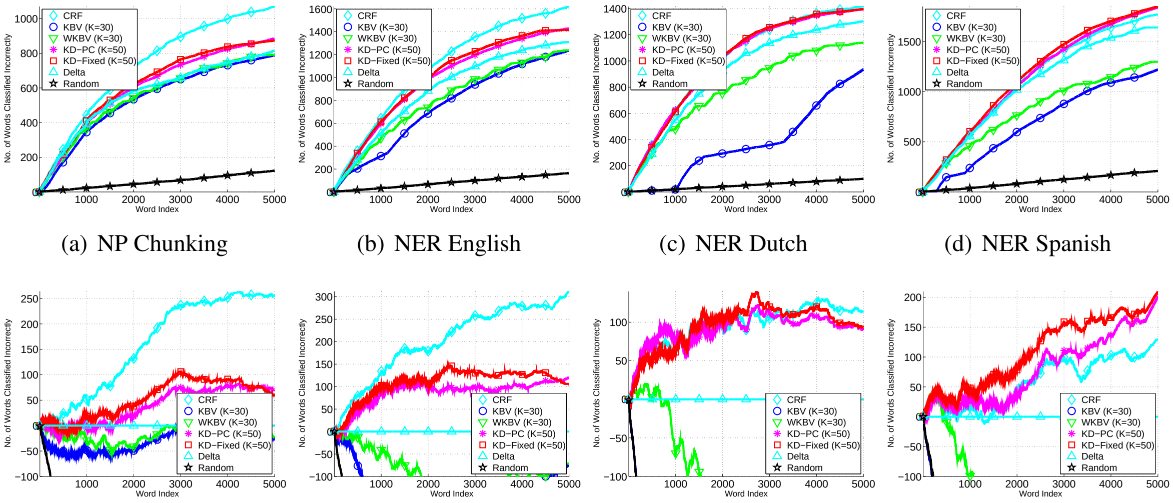
5 Confidence in the Prediction

Most large-margin-based training algorithms output models that their prediction is a single labeling of the input, with no additional confidence information about the correctness of that prediction. This situation is acceptable when the output of the system is used anyway, irrespectively of its quality. This situation is not acceptable when the output of the system is used as an input of another system that is sensitive to correctness of the specific prediction or that integrates various input sources. In such cases, additional confidence information about the correctness of these feeds for specific input can be used to improve the total output quality. Another case where such information is useful, is when there is additional agent that is validating the output of the system. The confidence information can be used to direct the check into small number of suspected predictions as opposed to random check, which may miss errors if their rate is small.

Some methods only provide *relative* confidence information. This information can be used to *rank* all predictions according to their confidence score, which can be used to direct a quality control component to detect errors in the prediction. Note, the confidence score is meaningless by itself and in fact, any monotonic transformation of the confidence scores yield equivalent confidence information. Other methods are providing confidence in the predicted output as an *absolute* information, that is, the probability of a prediction to be correct. We refer to these probabilistic outputs in a frequentists approach. When taking a large set of events (predictions) with similar probability confidence value ν of being correct, we expect that about ν fraction of the predictions in the group will be correct.

Algorithms: All of our methods to evaluate confidence, except two (Delta and CRF below), share the same conceptual approach and work in two stages. First, a method generates a set of K possible labelings for the input sentence (instead of a single prediction). Then, the confidence in a predicted labeling for a specific word is defined to be the proportion of labelings which are consistent with the predicted label. Formally, let $\boldsymbol{z}^{(i)}$ for $i = 1 \ldots K$ be the K labelings for some input $\boldsymbol{x}$, and let $\hat{\boldsymbol{y}}$ be the actual prediction for the input. (We do not assume that $\hat{\boldsymbol{y}} = \boldsymbol{z}^{(i)}$ for some i). The confidence in the label $\hat{y}_p$ of word $p = 1 \ldots |\boldsymbol{x}|$ is defined to be

$$\nu_p = |\{i \,:\, \hat{y}_p = z_p^{(i)}\}|/K \,. \qquad (5)$$



(a) NP Chunking (b) NER English (c) NER Dutch (d) NER Spanish

(e) NP Chunking (f) NER English (g) NER Dutch (h) NER Spanish

Figure 3: Total number of detected erroneous words vs. the number of ranked words (top panels), and relative to the Delta method (bottom panels). In other words, the lines in the bottom panels are the number of *additional* erroneous words detected compared to Delta method. All methods builds on the same weight-vector except CRF (see text).

We tried four approaches to generate the set of K possible labelings. The first method is valid only for methods that induce a probability distribution over predicted labels. In this case, we draw K labelings from this distribution. Specifically, we exploit the Gaussian distribution over weight vectors $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ maintained by AROW and CW, by inducing a distribution over labelings given an input. The algorithm samples K weight vectors according to this Gaussian distribution and outputs the best labeling with respect to each weight vector. Formally, we define the set $Z = \{\boldsymbol{z}^{(i)} : \boldsymbol{z}^{(i)} = \arg\max_{\boldsymbol{z}} \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{z}) \text{ where } \boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)\}$

The predictions of algorithms that use the mean weight vector $\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{z}} \boldsymbol{\mu} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{z})$ are invariant to the value of the input Σ (as noted by (Crammer et al., 2008)). However for the purpose of confidence estimation the specific value of Σ has a huge affect. Small eigenvalue of Σ yield that all the elements of Z will be the same, while large values yield random elements in the set, ignoring the input.

One possible simple option is to run the algorithm few times, with few possible initializations of Σ and choose one using the training set. However since the actual predictions of all these versions is the same (invariance to scaling, see (Crammer et al., 2008)) in practice we run the algorithm once initializing $\Sigma = I$. Then, after the training is completed, we try few scalings of the final covariance $s\Sigma$ for some positive scalar s, and choose the best value s using the training set. We refer to this method as KD-PC for K-Draws by Parameters Confidence.

The second method to estimate confidence follows the same conceptual steps, except that we used an isotropic covariance matrix, $\Sigma = sI$ for some positive scale information s. As before, the value of s was tuned on the training set. We denote this method KD-Fixed for K Draws by Fixed Standard Deviation. This method is especially appealing, since it can be used in combination with training algorithms that do *not* maintain confidence information, such as the Perceptron or PA.

Our third and fourth methods are deterministic and do not involve a stochastic process. We modified the Viterbi algorithm to output the K distinct labelings with highest score (computed using the mean weight vector in case of CW or AROW). The third method assigns uniform importance to each of the K labelings ignoring the actual score values. We call this method KBV, for K-best Viterbi. We thus propose the fourth method in which we define an importance weight ω_i to each labeling $\boldsymbol{z}^{(i)}$ and evaluate confidence using the weights, $\nu_p = \left(\sum_{i \text{ s.t. } \hat{y}_p = z_p^{(i)}} \omega_i\right) / \left(\sum_i \omega_i\right)$, where we set the weights to be their score value clipped at zero from below $\omega_i = \max\{0, \boldsymbol{\mu} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{z}^{(i)})\}$. (In practice, top score was always positive.) We call this method WKBV for weighted K-best Viterbi.

In addition to these four methods we propose a fifth method that is based on the margin and does not share the same conceptual structure of the previous methods. This method provide confidence score that is only relative and not absolute, namely its output can be used to compare the confidence in two labelings, yet there is no semantics defined over the scores. Given an input sentence to be labeled $\boldsymbol{x}$ and a model we define the confidence in the prediction associated with the pth word to be the difference in the highest score and the closest score, where we set the label of that word to anything but the label with the highest score. Formally, as before we define the best labeling $\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{z}} \boldsymbol{\mu} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{z})$, then the score of word p is defined to be, $\boldsymbol{\mu} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, \hat{\boldsymbol{y}}) - \max_{u \neq \hat{y}_p} \boldsymbol{\mu} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{z}_{|z_p=u})$, where we define the labeling $\boldsymbol{z}_{|z_p=u}$ to be the labeling that agrees with $\boldsymbol{z}$ on all words, except the pth word, where we define its label to be u. We refer to this method as Delta where the confidence information is a difference, aka as delta, between two score values.

Finally, as an additional baseline, we used a sixth method based on the confidence values for single words produced by CRF model. We considered the marginal probability of the word p to be assigned the predicted label $\hat{y}_p$ to be the confide value, this probability is calculated using the forward-backwards algorithm. This method is close in spirit to the Delta method as the later can be thought of computing marginals (in score, rather than probability). It also close to the K-Draws methods, as both CRF and K-Draws induce a distribution over labels. For CRF we can compute the marginals explicitly, while for the Gaussian models generated by CW (or AROW) the marginals can not be computed expliclity, and thus a sample based estimation (K-Draws) is used.

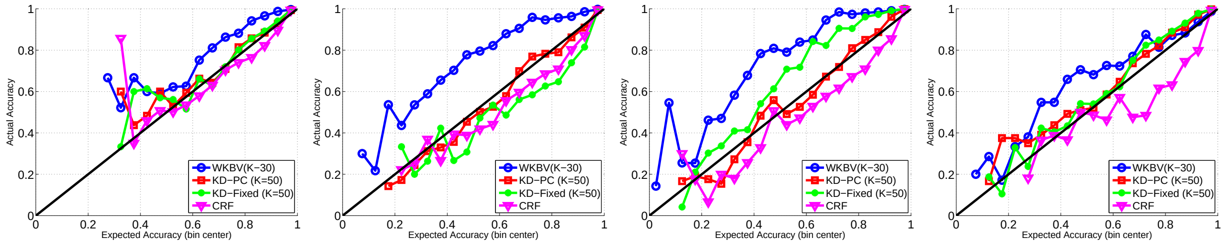
Experimental Setting: We evaluate the above methods as follows. We trained a classifier using the CW algorithm running for ten (10) iterations on three-fourth of the data and applied it to the remaining one-fourth to get a labeling of the test set. There are between $49K - 54K$ words to be labeled in all tasks, except NER Dutch where there are about $74K$ words. The fraction of words for which the trained model makes a mistake ranges between 2% (for NER Dutch) to 4.1% for NER Spanish.

We set the value of the hyper parameter ϕ to its optimal value obtained in the experiments reported in the previous section. The size of K of the number of labelings used in the four first methods (KD-PC, KD-Fixed, KBV, WKBV) and the weighting scalar s used in KD-PC and KD-Fixed were tuned for each dataset on a single evaluation on subset of the training set according to the best measured average precision. For the parameter s we tried about 20 values in the range 0.01 to 1.0, and for the number of labels K we tried the values in $10, 20 \ldots 80$. The optimal values are $K = 50$ for KD-PC and KD-Fixed, and $K = 30$ for KBV and WKBV. We noticed that KD-PC and KD-Fixed were robust to larger values of K, while the performance of KBV and WKBV was degraded significantly for large values of K.

We also trained CRF on the same training sets and applied it to label and assign confidence values to all the words in the test sets. The fraction of mislabeled words produced by the CRF model and the CW model is summarized in Table 3.

Relative Confidence: For each of the datasets, we first trained a model using the CW algorithm and applied each of the confidence methods on the output, ranking from low to high all the words of the test set according to the confidence in the prediction associated with them. Ideally, the top ranked words are the ones for which the classifier made a mistake on. This task can be thought of as a retrieval task of the erroneous words.

The average precision is the average of the precision values computed at all ranks of erroneous words. The average precision for ranking the words of the test-set according the confidence in the prediction of seven methods appears in the top-left panel of Fig. 2. (left to right bars in each group : CRF, KD-Fixed, KD-PC, Delta, WKBV, KBV and random ordering.) We see that when ordering the words randomly, the average precision is about the frequency of erroneous word, which is the lowest average precision. Next are the two methods based on the best Viterbi labelings, where the weighted approach outperforming the non-weighted version. Thus, taking the actual score value into consideration improves the ability to detect erroneous words. Next in performance is Delta, the margin-induced method. The



(a) NP Chunking (b) NER English (c) NER Dutch (d) NER Spanish

Figure 4: Predicted error in each bin vs. the actual frequency of mistakes in each bin. Best performance is obtained by methods close to the line $y = x$ (black line) for four tasks. Four methods are compared: weighted K-Viterbi (WKBV), K-draws PC (KD-PC) and K-draws fixed covariance (KD-Fixed) and CRF.

two best performing among the CW based methods are KD-Fixed and KD-PC, where the former is better in three out of four datasets. When compared to CRF we see that in two cases CRF outperforms the K-Draws based methods and in the other two cases it performs equally. We found the relative success of KD-Fixed compared to KD-PC surprising, as KD-Fixed does not take into consideration the actual uncertainty in the parameters learned by CW, and in fact replaced it with a *fixed* value across all features. Since this method does not need to assume a confidence-based learning approach we repeated the experiment, training a model with the passive-aggressive algorithm, rather than CW. All confidence estimation methods can be used except the KD-PC, which does take the confidence information into consideration. The results appear in the bottom-left panel of Fig. 2, and basically tell the same story, KD-Fixed outperform the margin based method (Delta), and the Viterbi based methods (KBV, WKBV).

To better understand the behavior of the various methods we plot the total number of detected erroneous words vs. the number of ranked words (first $5,000$ ranked words) in the top panels of Fig. 3. The bottom panels show the relative additional number of words each methods detects on top of the margin-based Delta method. Clearly, KD-Fixed and KD-PC detect erroneous words better than the other CW based methods, finding about 100 more words than Delta (when ranking $5,000$ words) which is about 8% of the total number of erroneous words.

Regarding CRF, it outperforms the K-Draws methods in NER English and NP chunking datasets, finding about 150 more words, CRF performed equally for NER Dutch, and performed worse for NER Spanish finding about 80 less words. We emphasize that all methods *except CRF* were based on the same exact weight vector, ranking the same predations, while CRF used an alternative weight vector that yields different number of erroneous words.

In details, we observe some correlation between the percentage or erroneous words in the entire set and the number of erroneous words detected among the first $5,000$ ranked words. For NP chunking and NER English datasets, CRF has more erroneous words compared to CW and it detects more erroneous words compared to K-Draws. For NER Dutch dataset CRF and CW have almost same number of erroneous words and almost same number of erroneous words detected, and finally in NER Spanish dataset CRF has fewer erroneous words and it detected less erroneous words. In other words, where there are more erroneous words to find (e.g. CRF in NP chunking), the task of ranking erroneous words is easier, and vice-versa.

We hypothesize that part of the performance differences we see between the K-Draws and CRF methods is due to the difference in the number of erroneous words in the ranked set.

This ranking view can be thought of marking suspected words to be evaluated manually by a human annotator. Although in general it may be hard for a human to annotate a single word with no need to annotate its close neighbor, this is not the case here. As the neighbor words are already labeled, and pretty reliably, as mentioned above.

Absolute Confidence: Our next goal is to evaluate how reliable are the *absolute* confidence values output by the proposed methods. As before, the confidence estimation methods (KD-PC, KD-Fixed, KBV, WKBV and CRF) were applied on the entire set of predicted labels. (Delta method is omitted as the confidence score it produces is not in $[0, 1]$).

For each of the four datasets and the five algorithms we grouped the words according to the value of their confidence. Specifically, we used twenty (20) bins dividing uniformly the confidence range into intervals of size 0.05. For each bin, we computed the fraction of words predicted correctly from the words assigned to that bin. Ultimately, the value of the computed frequency should be about the center value of the interval of the bin. Formally, bin indexed j contains words with confidence value in the range $[(j-1)/20, j/20)$ for $j = 1 \ldots 20$. Let b_j be the center value of bin j, that is $b_j = j/20 - 1/40$. The frequency of correct words in bin j, denoted by c_j is the fraction of words with confidence $\nu \in [(j-1)/20, j/20)$ that their assigned label is correct. Ultimately, these two values should be the same, $b_j = c_j$, meaning that the confidence information is a good estimator of the frequency of correct labels. Methods for which $c_j > b_j$ are too pessimistic, predicting too high frequency of erroneous labels, while methods for which $c_j < b_j$ are too optimistic, predicting too low frequency of erroneous words.

The results are summarized in Fig 4, one panel per dataset, where we plot the value of the center-of-bin b_j vs. the frequency of correct prediction c_j, connecting the points associated with a single algorithm. Four algorithms are shown: KD-PC, KD-Fixed, WKBV and CRF. We omit the results of the KBV approach - they were substantially inferior to all other methods. Best performance is obtained when the resulting line is close to the line $y = x$.

From the plots we observe that WKBV is too pessimistic as its corresponding line (blue square) is above the line $y = x$. CRF method is too optimistic, its corresponding line is below the line $y = x$. The KD-Fixed method is too pessimistic on NER-Dutch and too optimistic on NER-English. The best method is KD-PC which, surprisingly, tracks the line $x = y$ pretty closely. We hypothesis that its superiority is because it makes use of the uncertainty information captured in the covariance matrix Σ which is part of the Gaussian distribution.

Finally, these bins plots does not reflect the fact that different bins were not populated uniformly, the bins with higher values were more heavily populated. We thus plot in the top-right of Fig. 2 the root mean-square error in predicting the bin center value given by $\sqrt{\left(\sum_j n_j (b_j - c_j)^2\right) / \left(\sum_j n_j\right)}$, where n_j is the number of words in the jth bin. We observed a similar trend to the one appeared in the previous figure. WKBV is the least-performing method, then KD-Fixed and CRF, and then KD-PC which achieved lowest RMSE in all four datasets. Similar plot but when using PA for training appear in the bottom-right panel of Fig. 2. In this case we also see that KD-Fixed is better than WKBV, even though both methods were not trained with an algorithm that takes uncertainty information into consideration, like CW.

The success of KD-PC and KD-Fixed in evaluating confidence led us to experiment with using similar techniques for inference. Given an input sentence, the inference algorithm samples K times from the Gaussian distribution and output the best labeling according to each sampled weight vector. Then the algorithm predicts for each word the most frequent label. We found this method inferior to inference with the mean parameters. This approach differs from the one used by (Crammer et al., 2009a), as they output the most frequent labeling in a set, while the predicted label of our algorithm may not even belong to the set of predictions.

6 Active Learning

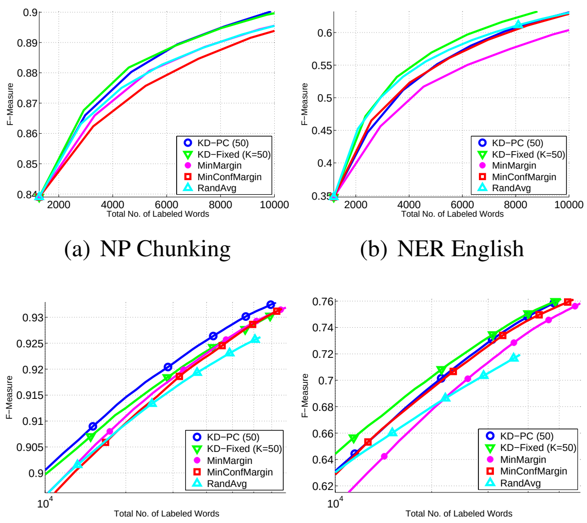
Encouraged by the success of the KD-PC and KD-Fixed algorithms in estimating the confidence in the prediction we apply these methods to the task of active learning. In active learning, the algorithm is given a large set of unlabeled data and a small set of labeled data and works in iterations. On each iteration, the overall labeled data at this point is used to build a model, which is then used to choose new subset of examples to be annotated.

In our setting, we have a large set of unlabeled sentences and start with a small set of 50 annotated sentences. The active learning algorithm is then using the CW algorithm to build a model, which in turn is used to rank sentences. The new data items are then annotated and accumulated to the set of labeled data points, ready for the next round. Many active learning algorithms are first computing a prediction for each of the unlabeled-data examples, which is then used to choose new examples to be labeled. In our case the goal is to label sentences, which are expensive to label. We thus applied the following setting. First, we chose a subset of 9K sentences as unlabeled training set, and another subset of size 3K for evaluation. After obtaining a model, the algorithm labels random $1,000$ sentences and chose a subset of 10 sentences using the active learning rule, which we will define shortly. After repeating this process 10 times we then evaluate the current model using the test data and proceed to choose new unlabeled examples to be labeled. Each method was applied to pick $5,000$ sentences to be labeled.

In the previous section, we used the confidence estimation algorithms to choose individual *words* to be annotated by a human. This setting is realistic since most words in each sentence were already classified (correctly). However, when moving to active learning, the situation changes. Now, all the words in a sentence are not labeled, thus a human may need to label additional words than the one in target, in order to label the target word. We thus experimented with the following protocol. On each iteration, the algorithm defines the score of an entire sentence to be the score of the least confident word in the sentence. Then the algorithm chooses the least confident sentence, breaking ties by favoring shorter sentences (assuming they contain relatively more informative words to be labeled than long sentences).

We evaluated five methods, KD-PC and KD-Fixed mentioned above. The method that ranks a sentence by the difference in score between the top- and second-best labeling, averaged over the length of sentence, denoted by MinMargin (Tong and Koller, 2001). A similar approach, motivated by (Dredze and Crammer, 2008), normalizes MinMargin score using the confidence information extracted from the Gaussian covariance matrix, we call this method MinConfMargin. Finally, We also evaluated an approach that picks random sentences to be labeled, denoted by RandAvg (averaged 5 times).

The averaged cumulative F-measure vs. number of words labeled is presented in Figs. 5,6. We can see that for short horizon (small number of sentences) the MinMargin is worse (in three out of four data sets), while MinConfMargin is worse in NP Chunking. Then there is no clear winner, but the KD-Fixed seems to be the best most of the time. The bottom panels show the results for more than $10k$ training words. Here, the random method performing the worst, while KD-PC and KD-Fixed are the best, and as shown in (Dredze and Crammer, 2008), MinConfMargin outperforming MinMargin.



(a) NP Chunking

(b) NER English

(c) NP Chunking

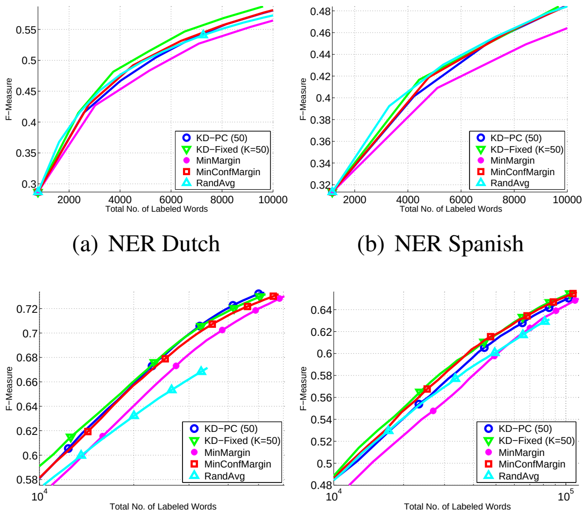
(d) NER English

Figure 5: Averaged cumulative F-score vs. total number of words labeled. The top panels show the results for up to $10,000$ labeled words, while the bottom panels show the results for more than $10k$ labeled words.

Related Work: Most previous work has focused on confidence estimation for an entire example or some fields of an entry (Culotta and McCallum, 2004) using CRFs. (Kristjansson et al., 2004) show the utility of confidence estimation is extracted fields of an interactive information extraction system by high-lighting low confidence fields for the user. (Scheffer et al., 2001) estimate confidence of single token label in HMM based information extraction system by a method similar to the Delta method we used. (Ueffing and Ney, 2007) propose several methods for word level confidence estimation for the task of machine translation. One of the methods they use is very similar to the weighted and non-weighted K-best Viterbi methods we used with the proper adjustments to the machine translation task.

Acknowledgments

The resrach is supported in part by German-Israeli Foundation grant GIF-2209-1912. KC is a Horev Fellow, supported by the Taub Foundations. The reviewers thanked for their constructive comments.



(a) NER Dutch

(b) NER Spanish

(c) NER Dutch

(d) NER Spanish

Figure 6: See Fig. 5

References

[Cesa-Bianchi and Lugosi2006] N. Cesa-Bianchi and G. Lugosi. 2006. *Prediction, Learning, and Games*. Cambridge University Press, New York, NY, USA.

[Collins2002] M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP*.

[Crammer et al.2005] K. Crammer, R. Mcdonald, and F. Pereira. 2005. Scalable large-margin online learning for structured classification. Tech. report, Dept. of CIS, U. of Penn.

[Crammer et al.2008] K. Crammer, M. Dredze, and F. Pereira. 2008. Exact confidence-weighted learning. In *NIPS 22*.

[Crammer et al.2009a] K. Crammer, M. Dredze, and A. Kulesza. 2009a. Multi-class confidence weighted algorithms. In *EMNLP*.

[Crammer et al.2009b] K. Crammer, A. Kulesza, and M. Dredze. 2009b. Adaptive regularization of weighted vectors. In *NIPS 23*.

[Culotta and McCallum2004] A. Culotta and A. McCallum. 2004. Confidence estimation for information extraction. In *HLT-NAACL*, pages 109–112.

[Dredze and Crammer2008] M. Dredze and K. Crammer. 2008. Active learning with confidence. In *ACL*.

[Dredze et al.2008] M. Dredze, K. Crammer, and F. Pereira. 2008. Confidence-weighted linear classification. In *ICML*.

[Kim et al.2000] E.F. Tjong Kim, S. Buchholz, and K. Sang. 2000. Introduction to the conll-2000 shared task: Chunking.

[Kristjansson et al.2004] T. Kristjansson, A. Culotta, P. Viola, and A. McCallum. 2004. Interactive information extraction with constrained conditional random fields. In *AAAI*, pages 412–418.

[Lafferty et al.2001] J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

[McCallum2002] Andrew McCallum. 2002. MALLET: A machine learning for language toolkit. `http://mallet.cs.umass.edu`.

[McDonald et al.2005a] R.T. McDonald, K. Crammer, and F. Pereira. 2005a. Flexible text segmentation with structured multilabel classification. In *HLT/EMNLP*.

[McDonald et al.2005b] Ryan T. McDonald, Koby Crammer, and Fernando C. N. Pereira. 2005b. Online large-margin training of dependency parsers. In *ACL*.

[Scheffer et al.2001] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. 2001. Active hidden markov models for information extraction. In *IDA*, pages 309–318, London, UK. Springer-Verlag.

[Sha and Pereira2003] Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proc. of HLT-NAACL*, pages 213–220.

[Shimizu and Haas2006] N. Shimizu and A. Haas. 2006. Exact decoding for jointly labeling and chunking sequences. In *COLING/ACL*, pages 763–770.

[Taskar et al.2003] B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin markov networks. In *nips*.

[Tjong and Sang2002] Erik F. Tjong and K. Sang. 2002. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *CoNLL*.

[Tjong et al.2003] E.F. Tjong, K. Sang, and F. De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *CoNLL*, pages 142–147.

[Tong and Koller2001] S. Tong and D. Koller. 2001. Support vector machine active learning with applications to text classification. In *JMLR*, pages 999–1006.

[Ueffing and Ney2007] Nicola Ueffing and Hermann Ney. 2007. Word-level confidence estimation for machine translation. *Comput. Linguist.*, 33(1):9–40.

[Wick et al.2009] M. Wick, K. Rohanimanesh, A. Culotta, and A. McCallum. 2009. Samplerank: Learning preferences from atomic gradients. In *NIPS Workshop on Advances in Ranking*.

Evaluating the Impact of Alternative Dependency Graph Encodings on Solving Event Extraction Tasks

Ekaterina Buyko and **Udo Hahn**

Jena University Language & Information Engineering (JULIE) Lab
Friedrich-Schiller-Universität Jena
Fürstengraben 30, 07743 Jena, Germany

{ekaterina.buyko|udo.hahn}@uni-jena.de

Abstract

In state-of-the-art approaches to information extraction (IE), dependency graphs constitute the fundamental data structure for syntactic structuring and subsequent knowledge elicitation from natural language documents. The top-performing systems in the BioNLP 2009 Shared Task on Event Extraction all shared the idea to use dependency structures generated by a variety of parsers — either directly or in some converted manner — and optionally modified their output to fit the special needs of IE. As there are systematic differences between various dependency representations being used in this competition, we scrutinize on different encoding styles for dependency information and their possible impact on solving several IE tasks. After assessing more or less established dependency representations such as the *Stanford* and *CoNLL-X* dependencies, we will then focus on trimming operations that pave the way to more effective IE. Our evaluation study covers data from a number of constituency- and dependency-based parsers and provides experimental evidence which dependency representations are particularly beneficial for the event extraction task. Based on empirical findings from our study we were able to achieve the performance of 57.2% F-score on the development data set of the BioNLP Shared Task 2009.

1 Introduction

Relation and event extraction are among the most demanding semantics-oriented NLP challenge tasks (both in the newspaper domain such as for ACE[1], as well as in the biological domain such as for BioCreative[2] or the BioNLP Shared Task[3]), comparable in terms of analytical complexity with recent efforts directed at opinion mining (e.g., NTCIR-7[4] or TREC Blog tracks[5]) or the recognition of textual entailment.[6] The most recent *BioNLP 2009 Shared Task on Event Extraction* (Kim et al., 2009) required, for a sample of 260 MEDLINE abstracts, to determine all mentioned events — to be chosen from a given set of nine event types, including *"Localization"*, *"Binding"*, *"Gene Expression"*, *"Transcription"*, *"Protein Catabolism"*, *"Phosphorylation"*, *"Positive Regulation"*, *"Negative Regulation"*, and (unspecified) *"Regulation"* — and link them appropriately with *a priori* supplied protein annotations. The demands on text analytics to deal with the complexity of this Shared Task in terms of relation diversity and specificity are unmatched by former challenges.

For relation extraction in the biomedical domain (the focus of our work), a stunning convergence towards dependency-based syntactic representation structures is witnessed by the performance results of the top-performing systems in the *BioNLP'09*

[1] http://papers.ldc.upenn.edu/LREC2004/ACE.pdf

[2] http://biocreative.sourceforge.net/

[3] www-tsujii.is.s.u-tokyo.ac.jp/GENIA/SharedTask/

[4] http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings7/pdf/revise/01-NTCIR-OV-MOAT-SekiY-revised-20081216.pdf

[5] http://trec.nist.gov/data/blog08.html

[6] http://pascallin.ecs.soton.ac.uk/Challenges/RTE/

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 982–992,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

Shared Task on Event Extraction.[7] Regarding the fact that dependency representations were always viewed as a vehicle to represent fundamental semantic relationships already at the syntactic level, this is not a great surprise. Yet, dependency grammar is not a monolithic, consensually shaped and well-defined linguistic theory. Accordingly, associated parsers tend to vary in terms of dependency pairing or structuring (which pairs of words join in a dependency relation?) and dependency typing (how are dependency relations for a particular pair labelled?).

Depending on the type of dependency theory or parser being used, various representations emerge (Miyao et al., 2007). In this paper, we explore these different representations of the dependency graphs and try, first, to pinpoint their effects on solving the overall event extraction task and, second, to further enhance the potential of JREX, a high-performance relation and event extractor developed at the JULIE Lab (Buyko et al., 2009).

2 Related Work

In the biomedical domain, the focus has largely been on binary relations, in particular protein-protein interactions (PPIs). Accordingly, the biomedical NLP community has developed various PPI-annotated corpora (e.g., LLL (Nédellec, 2005), AIMED (Bunescu et al., 2005), BIOINFER (Pyysalo et al., 2007)). PPI extraction does clearly not count as a solved problem, and a deeper look at its biological and representational intricacies is certainly worthwhile. The GENIA event corpus (Kim et al., 2008) and the BioNLP 2009 Shared Task data (Kim et al., 2009) contain such detailed annotations of PPIs (amongst others).

The BioNLP Shared Task was a first step towards the extraction of specific pathways with precise information about the molecular events involved. In that task, 42 teams participated and 24 of them submitted final results. The winner system, TURKU (Björne et al., 2009), achieved with 51.95% F-score the milestone result in that competition followed by the JULIELab system (Buyko et al., 2009) which peaked at 46.7% F-score. Only recently, an extension of the TURKU system, the TOKYO system, has been realized (Miwa et al., 2010). TOKYO system's event extraction capabilities are based on the TURKU system, yet TURKU's manually crafted rule system for post-processing and the combination of extracted trigger-argument relations is replaced by a machine learning approach in which rich features collected from classification steps for triggers and arguments are re-combined. TOKYO achieves an overall F-score of 53.29% on the test data, thus outperforming TURKU by 1.34 percentage points.

The three now top-performing systems, TOKYO, TURKU and JULIELab, all rely on dependency graphs for solving the event extraction tasks. While the TURKU system exploits the Stanford dependencies from the McClosky-Charniak parser (Charniak and Johnson, 2005), and the JULIELab system uses the CoNLL-like dependencies from the GDep parser (Sagae and Tsujii, 2007),[8] the TOKYO system overlays the Shared Task data with two parsing representations, *viz.* Enju PAS structure (Miyao and Tsujii, 2002) and GDep parser dependencies. Obviously, one might raise the question as to what extent the performance of these systems depends on the choice of the parser and its output representations. Miyao et al. (2008) already assessed the impact of different parsers for the task of biomedical relation extraction (PPI). Here we perform a similar study for the task of event extraction and focus, in particular, on the impact of various dependency representations such as Stanford and CoNLL'X dependencies and additional trimming procedures.

For the experiments on which we report here, we performed experiments with the JULIELab system. Our main goal is to investigate into the crucial role of proper representation structures for dependency graphs so that the performance gap from Shared Task results between the best-performing TOKYO system and the JULIELab system be narrowed.

3 Event Extraction

3.1 Objective

Event extraction is a complex task that can be subdivided into a number of subtasks depending on

[7] `www-tsujii.is.s.u-tokyo.ac.jp/GENIA/SharedTask/results/results-master.html`

[8] The GDep parser has been trained on the GENIA Treebank pre-official version of the version 1.0 converted with the script available from `http://w3.msi.vxu.se/~nivre/research/Penn2Malt.html`

whether the focus is on the event itself or on the arguments involved:

Event trigger identification deals with the large variety of alternative verbalizations of the same event type, e.g., whether the event is expressed in a verbal or in a nominalized form ("*A is expressed*" as well as "*the expression of A*" both refer to the same event type, *viz.* $Expression(A)$). Since the same trigger may stand for more than one event type, event trigger ambiguity has to be resolved as well.

Event trigger disambiguation selects the correct event name from the set of alternative event triggers.

Argument identification is concerned with finding all necessary participants in an event, i.e., the arguments of the relation.

Argument ordering assigns each identified participant its functional role within the event, mostly $Agent$ and $Patient$.

3.2 JULIELab System

The JULIELab solution can best be characterized as a single-step learning approach for event detection as the system does not separate the overall learning task into independent event trigger and event argument learning subtasks.[9] The JULIELab system incorporates manually curated dictionaries and machine learning (ML) methodologies to sort out associated event triggers and arguments on dependency graph structures. For argument extraction, the JULIELab system uses two ML-based approaches, a feature-based and a kernel-based one. Given that methodological framework, the JULIELab team scored on 2nd rank among 24 competing teams, with 45.8% precision, 47.5% recall and 46.7% F-score on all 3,182 events. After the competition, this system was updated and achieved 57.6% precision, 45.7% recall and 51.0% F-score (Buyko et al., 2010) using modified dependency representations from the MST parser (McDonald et al., 2005). In this study, we perform event extraction experiments with various dependency representations that allow us to measure their effects on the event extraction task and to increase the overall JULIELab system performance in terms of F-score.

4 Dependency Graph Representations

In this section, we focus on representation formats of dependency graphs. In Section 4.1, we introduce fundamental notions underlying dependency parsing and consider established representation formats for dependency structures as generated by various parsers. In Section 4.2, we account for selected trimming operations for dependency graphs to ease IE.

4.1 Dependency Structures: Representation Issues

Dependency parsing, in the past years, has increasingly been recognized as an alternative to long-prevailing constituency-based parsing approaches, particularly in semantically-oriented application scenarios such as information extraction. Yet even under purely methodologically premises, it has gained wide-spread attention as witnessed by recent activities performed as part of the "CoNLL Shared Tasks on Multilingual Dependency Parsing" (Buchholz and Marsi, 2006).

In a nutshell, in dependency graphs of sentences, nodes represent single words and edges account for head-modifier relations between single words. Despite this common understanding, concrete syntactic representations often differ markedly from one dependency theory/parser to the other. The differences fall into two main categories: dependency pairing or structuring (which pairs of words join in a dependency relation?) and dependency typing (how are dependency relations for a particular pair labelled?).

The CoNLL'X dependencies, for example, are defined by 54 relation types,[10] while the Stanford scheme (de Marneffe et al., 2006) incorporates 48 types (so called grammatical relations or Stanford dependencies). The Link Grammar Parser (Sleator and Temperley, 1991) employs a particularly fine-grained repertoire of dependency relations adding up to 106 types, whereas the well-known MINIPAR parser (Lin, 1998) relies on 59 types. Differences in dependency structure are at least as common as differences in dependency relation typing (see below).

[9] The JULIELab system considers all relevant lexical items as potential event triggers which might represent an event. Only those event triggers that can eventually be connected to arguments, finally, represent a true event.

[10] Computed by using the conversion script on WSJ data (accessible via `http://nlp.cs.lth.se/pennconverter/`; see also Johansson and Nugues (2007) for additional information). From the GENIA corpus, using this script, we could only extract 29 CoNLL dependency relations.

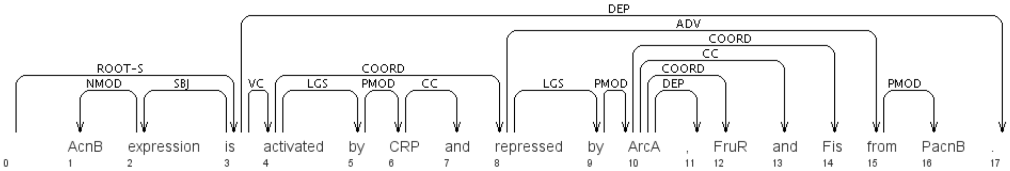


Figure 1: Example of CoNLL 2008 dependencies, as used in most of the native dependency parsers.

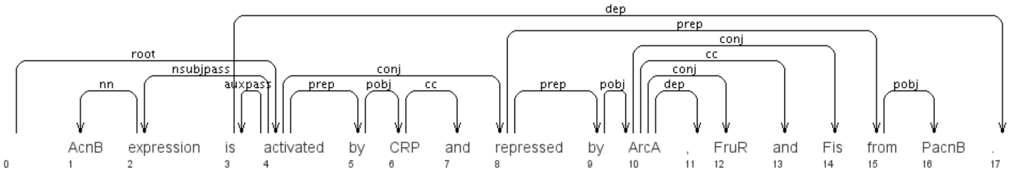


Figure 2: Stanford dependencies, *basic* conversion from Penn Treebank.

In general, dependency graphs can be generated by syntactic parsers in two ways. First, native dependency parsers output CoNLL'X or Stanford dependencies dependent on which representation format they have been trained on.[11] Second, in a derivative dependency mode, the output of constituency-based parsers, e.g., phrase structure representations, is subsequently converted either into CoNLL'X or Stanford dependencies using Treebank conversion scripts (see below). In the following, we provide a short description of these two established dependency graph representations:

- **CoNLL'X dependencies (CD).** This dependency tree format was used in the CoNLL'X Shared Tasks on multi-lingual dependency parsing (see Figure 1). It has been adopted by most native dependency parsers and was originally obtained from Penn Treebank (PTB) trees using constituent-to-dependency conversion (Johansson and Nugues, 2007). It differs slightly in the number and types of dependencies being used from various CoNLL rounds (e.g., CoNLL'08 provided a dependency type for representing appositions).[12]

- **Stanford dependencies (SD).** This format was proposed by de Marneffe et al. (2006) for semantics-sensitive applications using dependency representations, and can be obtained using the Stanford tools[13] from PTB trees. The Stanford format is widely used in the biomedical domain (e.g., by Miyao et al. (2008) or Clegg and Shepherd (2005)).

There are systematic differences between CoNLL'X and Stanford dependencies, e.g., as far as the representation of passive constructions, the position of auxiliary and modal verbs, or coordination representation is concerned. In particular, the representation of the *passive* construction and the role of the auxiliary verb therein may have considerable effects for semantics-sensitive tasks. While in SD the subject of the passive construction is represented by a special `nsubj` dependency label, in CD we find the same subject label as for active constructions `SUB(J)`. On CoNLL'08 data, the logical subject is marked by the `LGS` dependency edge that connects the passive-indicating preposition *"by"* with the logical subject of the sentence.

The representation of *active* constructions are similar in CD and SD though besides the role of auxiliary and modal verbs. In the Stanford dependency representation scheme, rather than taking auxiliaries to be the heads in passive or tense constructions, main verbs are assigned this grammatical function (see Figure 2). The CoNLL'X represen-

[11] We disregard in this study other dependency representations such as MINIPAR and LINK GRAMMAR representations.

[12] For the differences between CoNLL'07 and CoNLL'08 representations, cf. `http://nlp.cs.lth.se/software/treebank_converter/`

[13] Available from `nlp.stanford.edu/software/lex-parser.shtml`

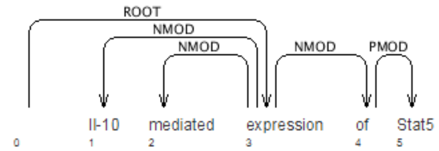


Figure 3: Noun phrase representation in CoNLL'X dependency trees.

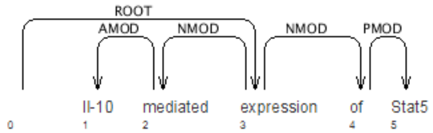


Figure 4: Trimming procedure *noun phrase* on CoNLL'X dependency trees.

tation scheme is completely different in that auxiliaries – much in common with standard dependency theory – are chosen to occupy the role of the governor (see Figure 1). From the perspective of relation extraction, however, the Stanford scheme is certainly closer to the desired predicate-argument structure representations than the CoNLL scheme.

4.2 Dependency Graph Modifications in Detail

Linguistic intuition suggests that the closer a dependency representation is to the format of the targeted semantic representation, the more likely will it support the semantic application. This idea is directly reflected in the Stanford dependencies which narrow the distance between nodes in the dependency graph by collapsing procedures (the so-called *collapsed* mode of phrase structure conversion). An example of collapsing is the conversion of "*expression* $\xrightarrow{\text{nmod}}$ *in* $\xrightarrow{\text{pmod}}$ *cells*" to "*expression* $\xrightarrow{\text{prep\_in}}$ *cells*". An extension of collapsing is the re-structuring of coordinations with sharing the dependency relations of conjuncts (the so-called *ccprocessed* mode of phrase structure conversion).

According to the Stanford scheme, Buyko et al. (2009) proposed collapsing scenarios on CoNLL'X dependency graphs. Their so-called *trimming* operations treat three syntactic phenomena, *viz.* coordinations (*coords*), auxiliaries/modals (*auxiliaries*), and prepositions (*preps*). For coordinations, they propagate the dependency relation of the first conjunct to all the other conjuncts within the coordination. For auxiliaries/modals, they prune the auxiliaries/modals as governors from the dependency graph and propagate the dependency relations of these nodes to the main verbs. Finally, for prepositions, they collapse a pair of typed dependencies into a single typed dependency (as illustrated above).

For the following experiments, we extended the trimming procedures and propose the re-structuring of noun phrases with action adjectives to make the dependency representation even more compact for semantic interpretation. The original dependency representation of the noun phrase selects the rightmost noun as the head of the NP and thus all remaining elements are its dependents (see Figure 3). For the noun phrases containing action adjectives (mostly verb derivations) this representation does not reflect the true semantic relations between the elements. For example, in *"IL-10 mediated expression"* it is *"IL-10"* that mediates the expression. Therefore, we re-structure the dependency graph by changing the head of *"IL-10"* from *"expression"* to *"mediated"*. Our re-coding heuristics selects, first, all the noun phrases containing action adjectives ending with *"-ed"*, *"-ing"*, *"-ible"* suffixes and with words such as *"dependent"*, *"specific"*, *"like"*. In the second step, we re-structure the noun phrase by encoding the adjective as the head of all the nouns preceding this adjective in the noun phrase under scrutiny (see Figure 4).

5 Experiments and Results

In this section, we describe the experiments and results related to event extraction tasks based on alternative dependency graph representations. For our experiments, we selected the following top-performing parsers — the first three phrase structure based and thus the origin of derivative dependency structures, the last three fully dependency based for making native dependency structures available:

- **C+J**, Charniak and Johnson's reranking parser (Charniak and Johnson, 2005), with the WSJ-trained parsing model.

- **M+C**, Charniak and Johnson's reranking parser (Charniak and Johnson, 2005), with the self-trained biomedical parsing model from McClosky (2010).

- **Bikel**, Bikel's parser (Bikel, 2004) with the WSJ-trained parsing model.

- **GDep** (Sagae and Tsujii, 2007), a native dependency parser.

- **MST** (McDonald et al., 2005), another native dependency parser.

- **MALT** (Nivre et al., 2007), yet another native dependency parser.

The native dependency parsers were re-trained on the GENIA Treebank (Tateisi et al., 2005) conversions.[14] These conversions,[15] i.e., Stanford *basic*, CoNLL'07 and CoNLL'08 were produced with the currently available conversion scripts. For the Stanford dependency conversion, we used the Stanford parser tool,[16] for CoNLL'07 and CoNLL'08 we used the treebank-to-CoNLL conversion scripts[17] available from the CoNLL'X Shared Task organizers.

The phrase structure based parsers were applied with already available models, i.e., the Bikel and C+J parsers as trained on the WSJ corpus, and M+C as trained on the GENIA Treebank corpus. For our experiments, we converted the prediction results of the phrase structure based parsers into five dependency graph representations, *viz.* Stanford *basic*, Stanford *collapsed*, Stanford *ccprocessed*, CoNLL'07 and CoNLL'08, using the same scripts as for the conversion of the GENIA Treebank.

The JULIELab event extraction system was re-trained on the Shared Task data enriched with different outputs of syntactic parsers as described above. The results for the event extraction task are represented in Table 1. Due to the space limitation of this paper we provide the summarized results of important event extraction sub-tasks only, i.e., results for basic events (*Gene Expression*, *Transcription*, *Localization*, *Protein Catabolism*) are summarized under SVT-TOTAL; regulatory events are summarized under REG-TOTAL; the overall extraction results are listed in ALL-TOTAL (see Table 1).

Obviously, the event extraction system trained on various dependency representations indeed produces truly different results. The differences in terms of F-score come up to 2.4 percentage points for the SVT-TOTAL events (cf. the MALT parser, difference between SD *basic* (75.6% F-score) and CoNLL'07 (78.0% F-score)), up to 3.6 points for REG-TOTAL (cf. the M+C parser, difference between SD *ccprocessed* (40.9% F-score) and CoNLL'07 (44.5% F-score)) and up to 2.5 points for ALL-TOTAL (cf. the M+C parser, difference between SD *ccprocessed* (52.8% F-score) and CoNLL'07 (55.3% F-score)).

The top three event extraction results on the development data based on different syntactic parsers results are achieved with M+C parser – CoNLL'07 representation (55.3% F-score), MST parser – CoNLL'08 representation (54.6% F-score) and MALT parser – CoNLL'08 representation (53.8% F-score) (see Table 1, ALL-TOTAL). Surprisingly, both the CoNLL'08 and CoNLL'07 formats clearly outperform Stanford representations on all event extraction tasks. Stanford dependencies seem to be useful here only in the *basic* mode. The *collapsed* and *ccprocessed* modes produce even worse results for the event extraction tasks.

Our second experiment focused on trimming operations on CoNLL'X dependency graphs. Here we performed event extraction after the trimming of the dependency trees as described in Section 4.2 in different modes: *coords* – re-structuring coordinations; *preps* – collapsing of prepositions; *auxiliaries* – propagating dependency relations of auxiliars and modals to main verbs; *noun phrase* – re-structuring noun phrases containing action adjectives. Our second experiment showed that the extraction of selected events can profit in particular from the trimming procedures *coords* and *auxiliaries*, but there is no evidence for a general trimming configuration for the overall event extraction task.

In Table 2 we summarize the best configurations we found for the events in focus. It is quite evident that the CoNLL'08 and CoNLL'07 dependencies modified for auxiliaries and coordinations are the best configurations for four events (out of nine). For three events no modifications are necessary and

[14] For the training of dependency parsers, we used from the available Stanford conversion variants only Stanford *basic*. The *collapsed* and *ccprocessed* variants do not provide dependency trees and are not recommended for training native dependency parsers.

[15] We used the GENIA Treebank version 1.0, available from `www-tsujii.is.s.u-tokyo.ac.jp`

[16] `http://nlp.stanford.edu/software/lex-parser.shtml`

[17] `http://nlp.cs.lth.se/software/treebank_converter/`

Parser	SD basic			SD collapsed			SD ccprocessed			CoNLL'07			CoNLL'08		
SVT-TOTAL															
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
Bikel	70.5	75.5	**72.9**	70.7	74.5	72.5	71.6	73.5	72.5	69.4	75.9	72.5	69.7	75.7	72.6
C+J	73.0	77.4	75.1	73.2	77.3	75.2	72.8	77.2	75.0	73.5	78.3	**75.8**	73.0	77.9	75.4
M+C	76.4	78.0	77.2	76.4	77.6	77.0	76.4	77.2	76.8	76.4	79.0	77.7	76.6	79.3	**77.9**
GDEP	77.1	77.5	**77.3**	N/A	N/A	N/A	N/A	N/A	N/A	72.5	80.2	76.1	72.6	77.2	74.8
MALT	73.1	78.2	75.6	N/A	N/A	N/A	N/A	N/A	N/A	75.9	80.3	**78.0**	73.7	78.2	75.9
MST	76.4	78.5	77.4	N/A	N/A	N/A	N/A	N/A	N/A	74.8	78.4	76.6	76.7	80.8	**78.7**
REG-TOTAL															
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
Bikel	35.3	40.6	**37.8**	33.8	40.3	36.8	34.3	39.6	36.8	33.9	39.2	36.3	34.0	41.0	37.2
C+J	36.2	41.8	38.8	37.3	41.8	39.4	36.5	41.9	39.0	38.1	43.9	**40.8**	37.4	44.0	40.4
M+C	39.4	45.5	42.3	38.8	45.3	41.8	38.5	43.7	40.9	41.9	47.4	**44.5**	40.1	47.9	43.7
GDEP	39.6	42.8	41.6	N/A	N/A	N/A	N/A	N/A	N/A	38.4	43.7	40.9	39.8	44.4	**42.0**
MALT	38.8	44.3	41.4	N/A	N/A	N/A	N/A	N/A	N/A	39.0	44.3	41.5	39.2	46.4	**42.5**
MST	39.5	43.6	41.4	N/A	N/A	N/A	N/A	N/A	N/A	39.6	45.6	42.4	40.6	45.8	**43.0**
ALL-TOTAL															
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
Bikel	47.4	51.5	**49.4**	46.3	50.8	48.5	46.9	50.2	48.5	44.8	50.7	47.6	44.7	51.8	48.0
C+J	49.3	53.8	51.5	49.6	52.8	51.2	49.0	53.0	50.9	50.3	54.4	**52.3**	49.5	54.3	51.8
M+C	52.3	56.4	54.3	51.8	55.7	53.7	51.3	54.3	52.8	53.2	57.5	**55.3**	52.2	58.2	55.0
GDEP	52.7	54.5	**53.6**	N/A	N/A	N/A	N/A	N/A	N/A	50.6	55.2	52.8	51.3	55.0	53.1
MALT	50.4	54.7	52.4	N/A	N/A	N/A	N/A	N/A	N/A	51.5	56.0	53.7	51.2	56.8	**53.8**
MST	52.3	54.8	53.5	N/A	N/A	N/A	N/A	N/A	N/A	51.7	56.4	53.9	52.4	56.9	**54.6**

Table 1: Results on the Shared Task development data for Event Extraction Task. Approximate Span Matching/Approximate Recursive Matching.

Event Class	Best Parser	Best Configuration	R	P	F
Gene Expression	MST	CoNLL'08, *auxiliaries, coords*	79.5	81.8	80.6
Transcription	MALT	CoNLL'07, *auxiliaries, coords*	67.1	75.3	71.0
Protein Catabolism	MST	CoNLL'08, *preps*	85.7	100	92.3
Phosphorylation	MALT	CoNLL'08	80.9	88.4	84.4
Localization	MST	CoNLL'08, *auxiliaries*	81.1	87.8	84.3
Binding	MST	CoNLL'07, *auxiliaries, coords, noun phrase*	51.2	51.0	51.1
Regulation	MALT	CoNLL'07, *auxiliaries, coords*	30.8	49.5	38.0
Positive Regulation	M+C	CoNLL'07	43.0	49.9	46.1
Negative Regulation	M+C	CoNLL'07	49.5	45.3	47.3

Table 2: Best Configurations for Dependency Representations for Event Extraction Task on the development data.

Binding	R	P	F
CoNLL'07	47.3	46.8	47.0
CoNLL'07 *auxiliaries, coords*	46.8	48.1	47.4
CoNLL'07 *auxiliaries, coords, noun phrase*	51.2	51.0	**51.1**

Table 3: Effects of trimming of *CoNLL* dependencies on the Shared Task development data for *Binding* events. Approximate Span Matching/Approximate Recursive Matching. The data was processed by the MST parser.

		JULIELab			JULIELab			TOKYO System		
		(M+C, CoNLL'08)			Final Configuration					
Event Class	gold	R	P	F	R	P	F	R	P	F
Gene Expression	356	79.2	80.3	79.8	79.5	81.8	80.6	78.7	79.5	79.1
Transcription	82	59.8	72.0	65.3	67.1	75.3	71.0	65.9	71.1	68.4
Protein Catabolism	21	76.2	88.9	82.0	85.7	100	92.3	95.2	90.9	93.0
Phosphorylation	47	83.0	81.2	82.1	80.9	88.4	84.4	85.1	69.0	76.2
Localization	53	77.4	74.6	75.9	81.1	87.8	84.3	71.7	82.6	76.8
SVT-TOTAL	559	76.4	79.0	**77.7**	78.2	82.6	**80.3**	77.3	77.9	**77.6**
Binding	248	45.6	45.9	45.8	51.2	51.0	51.1	50.8	47.6	49.1
EVT-TOTAL	807	66.9	68.7	**67.8**	69.9	72.5	**71.2**	69.1	68.1	**68.6**
Regulation	169	32.5	46.2	38.2	30.8	49.5	38.0	36.7	46.6	41.1
Positive_regulation	617	42.3	49.0	45.4	43.0	49.9	46.1	43.9	51.9	47.6
Negative_regulation	196	48.5	44.0	46.1	49.5	45.3	47.3	38.8	43.9	41.2
REG-TOTAL	982	41.9	47.4	**44.5**	42.2	48.7	**45.2**	41.7	49.4	**45.2**
ALL-TOTAL	1789	53.2	57.5	**55.3**	54.7	60.0	**57.2**	54.1	58.7	**56.3**

Table 4: Results on the Shared Task development data. Approximate Span Matching/Approximate Recursive Matching.

		JULIELab			JULIELab			TOKYO system		
		(Buyko et al., 2010)			Final Configuration					
Event Class	gold	R	P	F	R	P	F	R	P	F
Gene Expression	722	66.3	79.6	72.4	67.0	77.2	71.8	68.7	79.9	73.9
Transcription	137	33.6	61.3	43.4	35.0	60.8	44.4	54.0	60.7	57.1
Protein Catabolism	14	71.4	90.9	80.0	71.4	90.9	80.0	42.9	75.0	54.6
Phosphorylation	135	80.0	85.0	82.4	80.7	84.5	82.6	84.4	69.5	76.3
Localization	174	47.7	93.3	63.1	45.4	90.8	60.5	47.1	86.3	61.0
SVT-TOTAL	1182	61.4	80.3	**69.6**	61.8	78.2	**69.0**	65.3	76.4	**70.4**
Binding	347	47.3	52.4	49.7	47.3	52.2	49.6	52.2	53.1	52.6
EVT-TOTAL	1529	58.2	73.1	**64.8**	58.5	71.7	**64.4**	62.3	70.5	**66.2**
Regulation	291	24.7	40.5	30.7	26.8	38.2	31.5	28.9	39.8	33.5
Positive Regulation	983	35.8	45.4	40.0	34.8	45.8	39.5	38.0	48.3	42.6
Negative Regulation	379	37.2	39.7	38.4	37.5	40.9	39.1	35.9	47.2	40.8
REG-TOTAL	1653	34.2	43.2	**38.2**	34.0	43.3	**38.0**	35.9	46.7	**40.6**
ALL-TOTAL	3182	45.7	57.6	**51.0**	45.8	57.2	**50.9**	48.6	59.0	**53.3**

Table 5: Results on the Shared Task test data. Approximate Span Matching/Approximate Recursive Matching.

only one event profits from trimming of prepositions (*Protein Catabolism*). Only the *Binding* event profits significantly from noun phrase modifications (see Table 3). The increase in F-score for trimming procedures is 4.1 percentage points for *Binding* events.

In our final experiment we connected the best configurations for each of the BioNLP'09 events as presented in Table 2. The overall event extraction results of this final configuration are presented in Tables 4 and 5. We achieved an increase of 1.9 percentage points F-score in the overall event extraction compared to the best-performing single parser configuration (M+C, CoNLL'07) (see Table 4, ALL-TOTAL). The reported results on the development data outperform the results of the TOKYO system by 2.6 percentage points F-score for all basic events including *Binding* events (see Table 4, EVT-TOTAL) and by 0.9 percentage points in the overall event extraction task (see Table 4, ALL-TOTAL).

On the test data we achieved an F-score similar to the current JULIELab system trained on modified CoNLL'07 dependencies from the MST parser (see Table 5, ALL-TOTAL).[18] The results on the official test data reveal that the performance differences between various parsers may play a much smaller role than the proper choice of dependency representations.[19] Our empirical findings that the best performance results could only be achieved by event-specific dependency graph configurations reveal that the syntactic representations of different semantic events vary considerably at the level of dependency graph complexity and that the automatic prediction of such syntactic structures can vary from one dependency parser to the other.

6 Discussion

The evaluation results from Table 1 show that an increased F-score is basically due to a better performance in terms of precision. For example, the M+C evaluation results in the Stanford *basic* mode provide an increased precision by 2 percentage points compared to the Stanford *ccprocessed* mode. Therefore, we focus here on the analysis of false positives that the JULIELab system extracts in various modes.

For the first analysis we took the outputs of the systems based on the M+C parsing results. We scrutinized on the Stanford *basic* and *ccprocessed* false positives (fps) and we compared the occurrences of dependency labels in two data sets, namely the intersection of false positives from both system modes (set *A*) and the false positives produced only by the system with a worse performance (set *B*, *ccprocessed* mode). About 70% of all fps are contained in set *A*. Our analysis revealed that some dependency labels have a higher occurrence in set *B*, e.g., `nsubjpass, prep_on, prep_with, prep_in, prep_for, prep_as`. Some dependency labels occur only in set *B* such as `agent`, `prep_unlike, prep_upon`. It seems that collapsing some prepositions, such as *"with"*, *"in"*, *"for"*, *"as"*, *"on"*, *"unlike"*, *"upon"*, does not have a positive effect on the extraction of argument structures. In a second step, we compared the Stanford *basic* and CoNLL'07 false positive sets. The fps of both systems have an intersection of about 70%. We also compared the intersection of fps between two outputs (set *A*) and the set of additional fps of the system with worse results (Stanford *basic* mode, set *B*). The dependency labels such as `abbrev, dep, nsubj, nsubjpass` have a higher occurrence in set *B* than in set *A*. This analysis renders evidence that the distinction of `nsubj` and `nsubjpass` does not seem to have been properly learned for event extraction.

For the second analysis round we took the outputs of the MST parsing results. As in the previous experiments, we compared false positives from two mode outputs, here the CoNLL'07 mode and the CoNLL'07 modified for *auxiliaries* and *coordinations* mode. The fps have an intersection of 75%. The dependency labels such as `VC, SUBJ, COORD`, and `IOBJ` occur more frequently in the additional false positives from the CoNLL'07 mode than in the intersection of false positives from both system outputs. Obviously, the trimming of auxiliary and coordination structures has a direct positive effect on the argument extraction reducing false positive numbers especially with corresponding dependency labels in shortest dependency paths.

Our analysis of false positives shows that the distinction between active and passive subject labels,

[18] The current JULIELab system uses event-specific trimming procedures on CoNLL'07 dependencies determined on the development data set (see Buyko et al. (2010)).

[19] Trimmed CoNLL dependencies are used in both system configurations.

abbreviation labels, as well as collapsing prepositions in the Stanford dependencies, could not have been properly learned, which consequently leads to an increased rate of false positives. The trimming of auxiliary structures and the subsequent coordination collapsing on CoNLL'07 dependencies has indeed event-specific positive effects on the event extraction.

The main focus of this work has been on the evaluation of effects of different dependency graph representations on the IE task achievement (here the task of event extraction). But we also targeted the task-oriented evaluation of top-performing syntactic parsers. The results of this work indicate that the GENIA-trained parsers, i.e., M+C parser, the MST, MALT and GDep, are a reasonable basis for achieving state-of-the art performance in biomedical event extraction.

But the choice of the most suitable parser should also take into account its performance in terms of parsing time. Cer et al. (2010) and Miyao et al. (2008) showed in their experiments that native dependency parsers are faster than constituency-based parsers. When it comes to scaling event extraction to huge biomedical document collections, such as MEDLINE, the selection of a parser is mainly influenced by its run-time performance. MST, MALT and GDep parsers or the M+C parser with reduced reranking (Cer et al., 2010) would thus be an appropriate choice for large-scale event extraction under these constraints.[20]

7 Conclusion

In this paper, we investigated the role different dependency representations may have on the accomplishment of the event extraction task as exemplified by biological events. Different representation formats (mainly, Stanford *vs.* CoNLL) were then experimentally compared employing different parsers (Bikel, Charniak+Johnson, GDep, MST, MALT), both constituency based (for the derivative dependency mode) as well as dependency based (for the native dependency mode), considering different training scenarios (newspaper *vs.* biology domain).

From our experiments we draw the conclusion that the dependency graph representation has a crucial impact on the level of achievement of IE task requirements. Surprisingly, the CoNLL'X dependencies outperform the Stanford dependencies for four from six parsers. With additionally trimmed CoNLL'X dependencies we could achieve an F-score of 50.9% on the official test data and an F-score of 57.2% on the official development data of the BioNLP Shared Task on Event Extraction (see Table 5, ALL-TOTAL).

[20] For large-scale experiments an evaluation of the M+C with reduced reranking should be provided.

Acknowledgements

This research was partially funded by the BOOTSTREP project under grant FP6-028099 within the 6th Framework Programme (EC), by the CALBC project under grant FP7-231727 within the 7th Framework Programme (EC), and by the JENAGE project under grant 0315581D from the German Ministry of Education and Research (BMBF) as part of the GERONTOSYS funding initiative. We also want to thank Kenji Sagae (Institute for Creative Technologies, University of Southern California) for kindly providing the models of the GDEP parser.

References

Daniel M. Bikel. 2004. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4):479–511.

Jari Björne, Juho Heimonen, Filip Ginter, Antti Airola, Tapio Pahikkala, and Tapio Salakoski. 2009. Extracting complex biological events with rich graph-based feature sets. In *Proceedings BioNLP 2009. Companion Volume: Shared Task on Event Extraction*, pages 10–18. Boulder, Colorado, USA, June 4-5, 2009.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X Shared Task on multilingual dependency parsing. In *CoNLL-X – Proceedings of the 10th Conference on Computational Natural Language Learning*, pages 149–164, New York City, N.Y., June 8-9, 2006.

Razvan Bunescu, Ruifang Ge, Rohit J. Kate, Edward M. Marcotte, Raymond J. Mooney, Arun K. Ramani, and Yuk Wah Wong. 2005. Comparative experiments on learning information extractors for proteins and their interactions. *Artificial Intelligence in Medicine*, 33(2):139–155.

Ekaterina Buyko, Erik Faessler, Joachim Wermter, and Udo Hahn. 2009. Event extraction from trimmed dependency graphs. In *Proceedings BioNLP 2009. Companion Volume: Shared Task on Event Extrac-*

tion, pages 19–27. Boulder, Colorado, USA, June 4-5, 2009.

Ekaterina Buyko, Erik Faessler, Joachim Wermter, and Udo Hahn. 2010. Syntactic simplification and semantic enrichment - Trimming dependency graphs for event extraction. *Computational Intelligence*, 26(4).

Daniel Cer, Marie-Catherine de Marneffe, Dan Jurafsky, and Chris Manning. 2010. Parsing to Stanford Dependencies: Trade-offs between speed and accuracy. In *LREC'2010 – Proceedings of the 7th International Conference on Language Resources and Evaluation*, pages 1628–1632. Valletta, Malta, May 19-21, 2010.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine *n*-best parsing and MaxEnt discriminative reranking. In *ACL'05 – Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 173–180. Ann Arbor, MI, USA, 25-30, June, 2005.

Andrew B. Clegg and Adrian J. Shepherd. 2005. Evaluating and integrating Treebank parsers on a biomedical corpus. In *Proceedings of the ACL 2005 Workshop on Software*, pages 14–33. Ann Arbor, MI, USA, June 30, 2005.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC'2006 – Proceedings of the 5th International Conference on Language Resources and Evaluation*, pages 449–454. Genoa, Italy, 24-26 May 2006.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *NODALIDA 2007 – Proceedings of the 16th Nordic Conference of Computational Linguistics*, pages 105–112. Tartu, Estonia, May 24-25, 2007.

Jin-Dong Kim, Tomoko Ohta, and Jun'ichi Tsujii. 2008. Corpus annotation for mining biomedical events from literature. *BMC Bioinformatics*, 9(10).

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of BioNLP'09 Shared Task on Event Extraction. In *Proceedings BioNLP 2009. Companion Volume: Shared Task on Event Extraction*, pages 1–9. Boulder, Colorado, USA, June 4-5, 2009.

Dekang Lin. 1998. Dependency-based evaluation of MINIPAR. In *Proceedings of the LREC'98 Workshop on the Evaluation of Parsing Systems*, pages 48–56. Granada, Spain, 28-30 May 1998.

David McClosky. 2010. *Any Domain Parsing: Automatic Domain Adaptation for Natural Language Parsing*. Ph.D. thesis, Department of Computer Science, Brown University.

Ryan T. McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *HLT/EMNLP 2005 – Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing*, pages 523–530. Vancouver, B.C., Canada, October 6-8, 2005.

Makoto Miwa, Rune Sætre, Jin-Dong Kim, and Jun'ichi Tsujii. 2010. Event extraction with complex event classification using rich features. *Journal of Bioinformatics and Computational Biology*, 8:131–146.

Yusuke Miyao and Jun'ichi Tsujii. 2002. Maximum entropy estimation for feature forests. In *HLT 2002 – Proceedings of the 2nd International Conference on Human Language Technology Research*, pages 292–297. San Diego, CA, USA, March 24-27, 2002.

Yusuke Miyao, Kenji Sagae, and Jun'ichi Tsujii. 2007. Towards framework-independent evaluation of deep linguistic parsers. In *Proceedings of the GEAF 2007 Workshop*, CSLI Studies in Computational Linguistics Online, page 21 pages.

Yusuke Miyao, Rune Sætre, Kenji Sagae, Takuya Matsuzaki, and Jun'ichi Tsujii. 2008. Task-oriented evaluation of syntactic parsers and their representations. In *ACL 2008 – Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 46–54. Columbus, Ohio, USA, June 15-20, 2008.

Claire Nédellec. 2005. Learning Language in Logic: Genic interaction extraction challenge. In *Proceedings LLL-2005 – 4th Learning Language in Logic Workshop*, pages 31–37. Bonn, Germany, August 7, 2005.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2007. MALTPARSER: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Jarvinen, and Tapio Salakoski. 2007. BIOINFER: A corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 8(50).

Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *EMNLP-CoNLL 2007 – Proceedings of the Conference on Empirical Methods in Natural Language Processing and the Conference on Computational Natural Language Learning*, pages 1044–1050, Prague, Czech Republic, June 28-30, 2007.

Daniel Sleator and Davy Temperley. 1991. Parsing English with a link grammar. Technical report, Department of Computer Science, CMU.

Yuka Tateisi, Akane Yakushiji, and Jun'ichi Tsujii. 2005. Syntax annotation for the GENIA corpus. In *IJCNLP 2005 – Proceedings of the 2nd International Joint Conference on Natural Language Processing*, pages 222–227. Jeju Island, Korea, October 11-13, 2005.

Enhancing Mention Detection using Projection via Aligned Corpora

Yassine Benajiba
Center for Computational Learning Systems
Columbia University, NY
ybenajiba@ccls.columbia.edu

Imed Zitouni
IBM T.J. Watson Research Center
Yorktown Heights, NY
izitouni@us.ibm.com

Abstract

The research question treated in this paper is centered on the idea of exploiting rich resources of one language to enhance the performance of a mention detection system of another one. We successfully achieve this goal by projecting information from one language to another via a parallel corpus. We examine the potential improvement using various degrees of linguistic information in a statistical framework and we show that the proposed technique is effective even when the target language model has access to a significantly rich feature set. Experimental results show up to 2.4F improvement in performance when the system has access to information obtained by projecting mentions from a resource-rich-language mention detection system via a parallel corpus.

1 Introduction

The task of identifying and classifying entity textual references in open-domain texts, i.e. the *Mention Detection* (MD) task, has become one of the most important subtasks of Information Extraction (IE). It might intervene both as one step to structure natural language texts or as a text enrichment preprocessing step to help other Natural Language Processing (NLP) applications reach higher accuracy. Similarly to the Automatic Content Extraction (ACE) [1] nomenclature, we consider that a mention can be either named (e.g., John, Chicago), nominal (e.g., president, activist) or pronominal (e.g., he, she). It has also a specific class which describes the type of the entity it refers to. For instance, in the sentence:

Michael Bloomberg, the Mayor of NYC, declared his war on tobacco and sugary drinks in the city.

we find the mentions 'Michael Bloomberg', 'Mayor' and 'his' of the same person entity. Their types are named, nominal and pronominal, respectively. 'NYC' and 'city', on the other hand, are mentions of the same geopolitical (GPE) entity of type named and nominal, respectively. Consequently, MD is a more general and complex task than the well known Named Entity Recognition (NER) task which aims solely at the identification and classification of the named mentions.

The difficulty of the MD task is directly related to the nature of the language and the linguistic resources available, i.e. it is easier to build accurate MD systems for languages with a simple morphology and a high amount of linguistic resources. For this reason, we explore the idea of using an MD system, which has been designed and built for a resource-rich language (RRL), to help enhance the performance of an MD system in a target language (TL). More specifically, the goal of the research work we present in this paper is to employ the richness of English, in terms of natural language resources, to raise the accuracy of MD systems in other languages. For instance, an English MD system might achieve a performance of $F_{\beta=1}$-measure=82.7 (Zitouni and Florian, 2009) when it resorts to a rich set of features extracted from diverse resources, namely: part-of-speech, chunk information, syntactic parse trees, word sense information, WordNet information and information from the output of other mention detection classifiers. In this paper, our research question revolves around investigating an adequate approach to use such a system to the benefit of other languages such as Arabic, Chinese, French or Spanish MD systems, which also

[1] http://www.itl.nist.gov/iad/mig/tests/ace/2007/doc/ace07-evalplan.v1.3a.pdf

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 993–1001,
MIT, Massachusetts, USA, 9-11 October 2010.

have annotated resources but not of the same quantity and/or quality as English.

In this paper, we have targeted English and Arabic as the RRL and TL, respectively, because:

1. We have a very competitive English MD system;
2. The linguistic resources available for the Arabic language allow a simulation of different TL richness levels; and
3. The use of two languages of an utterly different nature makes the extrapolation of the results to other languages possible.

Our hypothesis might be expressed as follows: using an MD system resorting to a rich feature set (i.e. the RRL MD system) to boost a MD system performance in a TL can be very beneficial if the "donor" system surpasses its TL counterpart in terms of resources. To test this hypothesis, we have projected MD tags from RRL to TL via a parallel corpus, and then extracted several linguistic features about the automatically tagged words. Thereafter, we have conducted experiments adding these new features to the TL baseline MD system. In order to have a complete picture on the impact of these new features, we have used TL baseline systems resorting to a varied amount of features, starting with a case employing only lexical information to a case where we use all the resources we could gather for the TL. Experiments show that the gain is always statistically significant and it reaches its maximum when only very basic features are used in the baseline TL MD system.

2 Mention Detection

Similarly to classical NLP tasks, such as Base Phrase Chunking (Ramshaw and Marcus, 1999) (BPC) or NER (Tjong Kim Sang, 2002), we formulate the MD task as a sequence classification problem, i.e. the classifier assigns to each token in the text a label indicating whether it starts a specific mention, is inside a specific mention, or is outside any mentions. It also assigns to every non outside mention a class to specify its type: e.g., person, organization, location, etc. In this study, we chose the Maximum Entropy Markov Model (MEMM henceforth) approach because it can easily integrate arbitrary types of information in order to make a classification decision. To train our models, we have used the *Sequential Conditional Generalized Iterative Scaling* (SCGIS) technique (Goodman, 2002). This techniques uses a *Gaussian prior* for regularization (Chen and Rosenfeld, 2000). The features used by our MD systems can be divided into the following categories:

1- *Lexical*: these are token n-grams directly neighboring the current token on both sides, i.e. left and right. Empirical results have shown that the optimal span is $n = 3$.

2- *Syntactic*: they consist of the outcomes of several Part-Of-Speech (POS) taggers and BPCs trained on different corpora and different tag-sets in order to provide the MD system with a wider variety of information. Our model uses the POS and BPC information appearing in window of 5 (current, two previous, and two next) jointly with the tokens.

Both the English and the Arabic MD systems have access to lexical and syntactic features. The former one, however, also employs a set of features obtained from the output of other MD classifiers. In order to provide the MD system with complementary information, these classifiers are trained on different datasets annotated for different mention types, e.g. dates or occupation references (not used in our task).

3 Annotation, Projection and Feature Extraction

We remind the reader that our main goal is to use an RRL MD system to enhance the performance of an MD system in another language, i.e. the TL. In order to achieve this goal, we propose an approach that uses an RRL-to-TL parallel corpus to bridge between these two languages. This approach performs in three main steps, namely: annotation, projection and feature extraction. In this section, we describe in details each of these steps.

3.1 Annotation

This first step consists of MD tagging of the RRL side of the parallel corpus. Because in our case study we have chosen English as the RRL, we have used an accurate English MD system to perform the annotation step. Our English MD system achieves an F-measure of 82.7 (Zitouni and Florian, 2009) and has achieved significantly competitive results at the ACE evaluation campaign.

3.2 Projection

Once the RRL side of the parallel corpus is accurately augmented with MD tags, the projection step comes to transfer those tags to the TL side, Arabic in our case study, using the word alignment information. We illustrate the projection step with a relevant example. Let consider the following MD tagged English sentence:

*Bill/***B-PER-NAM** *Clinton/***I-PER-NAM** *is visiting North/***B-GPE-NAM** *Korea/***I-GPE-NAM** *today*

where "*Bill Clinton*" is a named person mention and "*North Korea*" is a named geopolitical entity (GPE) one. A potential Arabic translation of this sentence would be:

بيل كلينتون يزور كوريا الشمالية اليوم

which might be transliterated as:

byl klyntwn yzwr kwryA Al$mAlyA Alywm

After projecting the English mentions to the Arabic text, we obtain the following:

*byl/***B-PER-NAM** *klyntwn/***I-PER-NAM** *yzwr kwryA/***B-GPE-NAM** *Al$mAlyp/***I-GPE-NAM** *Alywm*

This tagged version of the Arabic text is provided to the third module of the process responsible on feature extraction (see Subsection 3.3). It is, however, pertinent to point out that the example we have used for illustration is relatively simple in the sense that almost all English and Arabic words have a 1-to-1 mapping. In real world translation (both human and automatic), one should expect to see 1-to-n, n-to-1 mappings as well as unmapped words on both sides of the parallel corpus rather frequently.

As stated by (Klementiev and Roth, 2006), the projection of NER tags is easier in comparison to projecting other types of annotations such as POS-tags and BPC[2], mainly because:

1. *Not all the words are mentions*: once we have projected the tags of the mentions from the RRL to TL side, the rest of tokens are simply considered as outside any mentions. This is different from the POS-tag and BPC where all the words are assigned a tag and thus when a word is unmapped, further processing is required (Yarowsky et al., 2001);

2. *In case of a 1-to-n mapping, the target n words are assigned the same class*: for instance, let consider the English GPE named mention "North-Korea". The segmented version of its Arabic translation would be "كوريا ال شمالية" (kwrya Al $mAlyp). The projection process consists in simply assigning the same class, i.e. GPE, to all Arabic tokens. The problem takes another dimension, however, in the case of propagating the POS-tags, because "North" is a NNP aligned with the determinant (DET) "Al" and the NNP "$mAlyp". Additional processing is needed to handle this difference of tags on the two sides.

3. *In case of n-to-1 mapping, the TL side word is simply assigned the class propagated from the RRL side*. For instance, if on the English side we have the named person multi-word mention "Ben Moussa", translated into the one-word mention بنموسى (bnmwsY) on the Arabic side, then projection consists of simply assigning the person named tag to the Arabic word.

However, in our research study, new challenges arose because our RRL data are automatically annotated, which is different from what has been reported in the research works we have mentioned before, i.e. (Yarowsky et al., 2001) and (Klementiev and Roth, 2006), where gold annotated data were used. In order to relax the impact of the noise introduced by the English MD system, we :

1. *use mention "splits" to filter annotation errors:* We assume that when a sequence of tokens is tagged as a mention on the RRL side, its TL counterpart should be an uninterrupted sequence of tokens as well. When the RRL MD system captures incorrectly the span of a mention, e.g. in the sentence "*Dona Karan international reputation of ...*", the RRL MD system might mistakenly tag "Dona Karan international" as an organization mention instead of tagging "Dona Karan" as a person mention. It is possible to detect this type of errors on the TL side because "dwnA kArAn" (Dona Karan) is distant from "Al EAlmyp" (international), i.e. they do not form an uninterrupted token sequence. We use this "split" in the mentions as information in order to not use these mentions in the feature extraction step (see Subsection 3.3).

2. *do not use the projected mentions directly for training:* Instead, we use these tags as additional features to our TL baseline model and allow our MEMM classifier to weigh them according to their relevance to each mention type.

3.3 Feature Extraction

At this point, the parallel corpus should be annotated with mentions on both of its sides. Where the RRL side is tagged using the English MD system during the annotation step (c.f section 3.1) while the TL side is annotated by the propagation of these MD tags via the parallel corpus in the projection step (c.f. section 3.2). In this third step, the goal is to extract pertinent linguistic features of the automatically tagged TL corpus to enhance MD model in the TL. The explored features are as follows:

[2] The claim is also valid for MD because it is the same type of annotation.

1. **Gazetteers:** we group mentions by class in different dictionaries. During both training and decoding, when we encounter a token or a sequence of tokens that is part of a dictionary, we fire its corresponding class; the feature is fired only when we find a complete match between sequence of tokens in the text and in the dictionary.
2. **Model-based features:** it consists of building a model on the automatically tagged TL side of the parallel corpus. The output of this model is used as a feature to enhance MD model in the target language. However, it is also possible to use this model to directly tag text in the TL. This would be useful in cases where we do not have any TL annotated data.
3. **n-gram context features:** it consists of using the annotated corpus in the TL to collect n-gram tokens surrounding a mention. We organize those contexts by mention type and we use them to tag tokens which appear in the same context in both the training and decoding sets. These tags will be used as additional feature in the MD model. For instance, if we consider that the person mention صدام حسين (SdAm Hsyn - Sadam Husein) appears in the following sentence: صرح أمس أن صدام حسين يترأس نظاما فاشلا which might be transliterated as: *SrH Ams An SdAm Hsyn ytrAs nZAmA fA$lA* and translated to English as: *declared yesterday that Sadam Husein governs a failed system*
the context n-grams that would be extracted are:

. **Left n-grams:** W_{-1}=أن (An - that), W_{-2}=أمس أن (Ams An - yesterday that), etc.

. **Right n-grams:** W_{+1}=يترأس (ystrAs - governs), W_{+2}=يترأس نظاما (ytrAs nZAmA - governs a system), etc.

. **Left and right n-grams:** a joint of the two previous features, W_{-i} and W_{+i}.

For both training and test data we create a new feature stream where we indicate that a token sequence is a mention if it appears in the same n-gram context.

4. **Head-word based features:** it considers that the lexical context in which the mention appeared is the sequence of the parent sub-trees head words in a parse-tree. For instance, if we consider the sentence which we have used in the previous example, the corresponding parse tree is shown in Figure 1.

The parent sub-tree heads of 'SdAm Hsyn' are marked with h_i on the tree. Similarly to the other features, in both training and decoding sets, we create a new feature stream where we tag those token sequences which appear with the same n first parent sub-tree head words as a person mention in the annotated TL data.

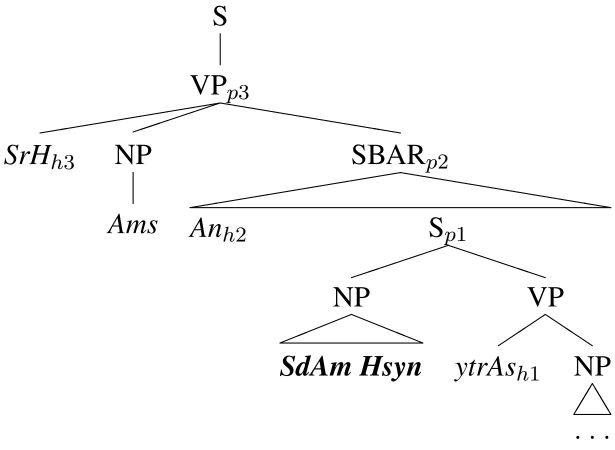


Figure 1: Parse tree

5. **Parser-based features:** it attempts to use the syntactic environment in which a mention might appear. In order to do so, for each mention in the target language corpus we consider only labels of the parent non-terminals .We mark parent non-terminal labels of 'SdAm Hsyn' on the tree with p_i. Similarly to the features described above, we create during both training and test a new feature stream where we indicate the token sequences which appear in the same parent non-terminal labels.

Gazetteers and model-based features are the most natural and expected kind of features that one would extract from the automatically MD tagged version of the TL text. Our motivation of using n-gram context features, on one hand, and the head-word based and parse-based features on the other is to: (i) contrast the impact of local and global context features; and (ii) experiment the possibility of employing both of them jointly in order to test their complementarity.

4 The Target Language Mention Detection System

- The Arabic language: In our research study, we have intentionally chosen a TL which is differs from English in its strategy in forming words and sentences. By doing so, we are seeking to avoid obtaining results which are biased by the similarity of the employed languages. For this reason, we have chosen Arabic as a TL.

Due to its Semitic origins, the Arabic language is both *derivational*, i.e. it uses a templatic strategy to form a word, and *highly inflectional*, i.e. additional affixes might be added to a word in order to obtain further meaning. Whereas the former characteristic is common in most languages, the latter, however, results in increasing *sparseness* in data and consequently forming an obstacle to achieve a high performance for most of the NLP tasks (Diab et al., 2004; Benajiba et al., 2008; Zitouni et al., 2005; Zitouni and Florian, 2008). From a NLP viewpoint, especially the supervised tasks such as the one we are dealing with in this paper, this implies that a huge amount of training data is necessary in order to build a robust model. In our study, to tackle the data sparseness problem, we have performed the *word segmentation*. This segmentation pre-processing step consists of separating the normal white-space delimited words into prefixes, stems, and suffixes. Thus, from a modeling viewpoint, the unit of analysis becomes the segments. We use a technique similar to the one introduced in (Lee et al., 2003) for segmentation with an accuracy of 98%.

- The Arabic MD system: Our Arabic MD system employs the same technique presented in Section 2. Compared to English MD model, Arabic MD system has access to morphological information (Stem) as we will explain next. Features used by the Arabic MD system are divided in three categories:

1. Lexical: Similar to the lexical features used by our English MD system (c.f. section 2);

2. Stem: This feature has been introduced in (Zitouni et al., 2005) as stem n-grams spanning the current stem; both preceding and following it. If the current token x_i is a stem, stem n-gram features contain the previous $n-1$ stems and the following $n-1$ stems. Stem n-gram features represent a lexical generalization that reduce data sparseness;

3. Syntactic: it consists of the output of POS taggers and the BPCs.

As we describe with more details in the experiments section (see Section 6), once we have extracted the new features from the parallel corpus, we contrast their impact with the level of richness in features of the TL MD system, i.e. we measure the impact of each feature f_i when the TL MD system uses: (i) only lexical features; (ii) both lexical and stem features; and (iii) lexical, stem and syntactic features.

5 Evaluation Data

Experiments are conducted on the Arabic ACE 2007 data. There are 379 Arabic documents and almost $98,000$ words. We find seven classes of mentions: Person (PER), Organization (ORG), Geo-Political Entity (GPE), Location (LOC), Facility (FAC), Vehicle (VEH) and Weapon (WEA). Since the evaluation test sets are not publicly available, we have split the publicly available *training* corpus into an 85%/15% data split. We use 323 documents ($80,000$ words) for training and 56 documents ($18,000$ words) as a test set. This results in $17,634$ mentions ($7,816$ named, $8,831$ nominal and 987 pronominal) for training and $3,566$ for test ($1,673$ named, $1,682$ nominal and 211 pronominal). To facilitate future comparisons with work presented here, and to simulate a realistic scenario, the splits are created based on article dates: the test data is selected as the latest 15% of the data in chronological order, in each of the covered genres (newswire and webblog). Performance on the ACE data is usually evaluated using a special-purpose measure, i.e. the ACE value metric. However, given that we are interested in the mention detection task only, we decided to use the more intuitive and popular (un-weighted) F-measure, the harmonic mean of precision and recall.

6 Experiments and Results

As we have stated earlier, our main goal is to investigate how an MD model of a TL might benefit from additional information about the mentions obtained by propagation from an RRL. In our research study we have chosen Arabic as the TL and English as the RRL. The English MD system we use has access to a large set of information (Zitouni and Florian, 2009) and has achieved a performance of 82.7F on ACE'07 data. In order to simulate different levels of resource-richness for the TL, we have employed four baseline systems which use different feature-sets. Following we present these feature-sets ranked from the resource-poorest to the resource-richest one: 1- $Lex.$: lexical features; 2- $Stem.$: $Lex.$ + stem features; and 3- $Syntac.$: $Stem.$ + syntactic features.

For each of these baseline systems, we study the impact of features extracted from the parallel corpus (c.f. Section 3) separately. We report the following results:

1- $Base.$: baseline system *without* the use of parallel-data extracted features;

2- $n-Lex.$: $Base.+$ n-gram context features;

	$Lex.$	$Stem$	$Syntac$
$Base.$	74.14	74.47	75.53
$n - Lex.$	74.71	75.25	76.20
$n - Head$	74.63	75.29	75.93
$n - Pars.$	75.32	75.19	75.74
Gaz	74.90	74.79	75.66
$Model$	74.60	75.50	76.22
$Comb.$	**76.01**	**76.74**	**77.18**

Table 1: Obtained results when the features were extracted from a hand-aligned parallel corpus

3- $n - Head$: $Base.$ + head-word based features;
4- $n - Pars.$: $Base.$ + parser-related features;
5- $Gaz.$: $Base.$ + automatically extracted gazetteers from the parallel corpus;
6- $Model$: $Base.$ + output of model trained on the Arabic part of the parallel corpus;
7- $Comb.$: combination of all the above.
In the rest of the paper, to measure whether the improvement in performance of a system using features from parallel data over baseline is statistically significant or not, we use the stratified bootstrap re-sampling significance test (Noreen, 1989) used in the NER shared task of CoNLL-2002[3]. We consider results as statistically significant when $p < 0.02$.

6.1 Hand-aligned Data

In our first experiment-set, we use a hand-aligned English-to-Arabic parallel corpus of approximately one million words. After tagging the Arabic side by projection we obtain 86.5K mentions. As we have previously mentioned, in order to generate the model-based feature, $Model$, we have trained a model on the Arabic side of the parallel corpus. This model achieved an F-measure of 57.7F. This shows the performance that might be achieved when *no human annotated data* is available in the TL.

Results in Table 1 show that a significant improvement is obtained when the TL is poor in resources; for instance an improvement of $\sim$1.9 points was achieved when the TL used only lexical features. The use of $n - Pars.$ features alone yielded 1.2 points of improvement. when the TL model uses a rich feature-set, we still can obtain $\sim$1.7 points improvement. When the TL baseline model employs the $Syntac$ feature-set, the greatest improvement is obtained when we add the model-based feature. Improvement obtained by the system using $Comb.$ features is statistically significant compared to the baseline model. This system also outperforms systems using the new feature set separately across the board. According to our error-analysis, the significant amount of Arabic mentions observed in the parallel corpus, where many of them do not appear in the training corpus, has significantly helped the $Lex.$, $Stem$ and $Syntac$ MD models to capture new mentions and/or correct the type assigned. Some of the relevant examples in our data are: (i) the facility mention مبنى بلفور (mbnY blfwr - Belvoir Building); (ii) the GPE mention كابول (kAbwl - Kabul); and (iii) the person mention البعثيّين (AlbEvyyn - the Baathists). These mentions have only been tagged correctly when we have added the new extracted features to our model.
In other words, the error-analysis clearly points out that one possible way to get further improvement is to increase the parallel data in order to increase the number of matches between (1) *the number of mentions which are wrongly tagged by the TL MD model* and (2) *the number of mentions in the TL side of the parallel corpus*. The second parameter can be, indirectly, increased by increasing the size of the parallel data. Getting 10 or 20 times more of parallel data that is hand-aligned is expensive and requires several months of human/hours work. For this reason we opted for using an unsupervised approach by selecting a parallel corpus that is automatically aligned as we discuss in the next section.

6.2 Automatically-aligned Data

We have used for this experiment-set an Arabic-to-English parallel data of 22 million words. The data in this corpus is automatically aligned using a technique presented in (Ittycheriah and Roukos, 2005). The alignment is one-to-many with a performance around 87 F-measure.

Because we are dealing with a large amount of data and the word alignment is done automatically, meaning more noise, we have used the English MD model confidence for additional filtering. Such filtering consists in keeping, from the parallel corpus, only sentences which have all tokens tagged with a confidence greater than α. In this paper, we use a value of $\alpha = 0.94$, which results in a corpus of 17 million words. We notice that a lower value of α results in a radical increase in noise. Because of space limitation, we will report results only with this value of α.

Table 2 shows the obtained results for parallel-

[3] http://www.cnts.ua.ac.be/conll2002/ner/

	$Lex.$	$Stem$	$Syntac$
$Base.$	74.14	74.47	75.53
$n - Lex.$	74.27	74.74	75.24
$n - Head.$	74.07	74.95	75.33
$n - Pars.$	75.62	75.22	76.02
Gaz	73.96	74.11	74.94
$Model$	74.87	75.12	75.76
$Comb.$	**75.56**	**75.93**	**76.46**

Table 2: Obtained results when the features were extracted from a automatically-aligned parallel corpus

data based features using the 17M subset. Differently from experiments using hand-aligned data, the best results have been obtained when we have used the parser-based feature, i.e. $n - Pars.$ On one hand, the overall behavior is comparable to the one obtained when using the 1M hand-aligned parallel data (see Table 1), i.e. (i) the greatest improvement has been obtained when the TL uses a poor feature-set; and (ii) when the TL baseline model is rich in resources, we still obtain 0.45 points absolute improvement when using $n - Pars.$ On the other hand, features extracted from automatically-aligned data, in comparison with the ones extracted from the hand aligned data, have helped the MD model to correct many of the TL baseline model false negatives. This has been observed when the TL baseline system uses a rich feature set as well. A side effect of the noisy word alignment, however, was an increase in the number of false positives. For instance, the word مستحضرات (mstHDrAt - preparations) which appeared in the following sentence:

عدم السماح لمستحضرات أخرى

which might be transliterated as:

Edm AlsmAH lmstHDrAt AxrY

and translated to English as:

not to allow other preparations

has been tagged as an organization mention because it has been mistakenly aligned, in the parallel corpus, with the word كاو, *KO*, in the sentence:

شركة كاو الكبرى للّمستحضرات التجميلية

meaning:

The big cosmetics company KO.

In order to validate our results, we run our experiments on a blind test-set. We have selected the latest 5% of each genre of the hand-aligned data and they have been manually annotated by a human. The blind test-set consists of 51,781 tokens of which 6,481 are mentions. Table 3 shows the distribution of these mentions over the different classes. The results are shown in Table 4. These results confirm the conclusions we have deduced from the ones previously presented in Table 2, i.e.: (i) the highest improvement is obtained when the TL is resource-scarce.

Class	Num. of mentions
FAC	285
GPE	2,145
LOC	239
ORG	1,135
PER	2,474
VEH	65
WEA	138

Table 3: Distribution over the classes of the blind test mentions

	$Lex.$	$Stem$	$Syntac$
$Base.$	74.26	73.54	73.61
$n - Lex.$	74.04	73.72	73.83
$n - Head$	74.14	73.64	73.83
$n - Pars.$	74.32	74.18	74.32
Gaz	71.49	72.13	73.39
$Model$	**75.01**	**74.66**	**74.78**

Table 4: Obtained results on blind test

6.3 Combining Hand-aligned and Automatically-aligned Data

Table 5 shows that combining both features extracted from hand-aligned and automatically-aligned corpora has led to better results. The im-

	$Lex.$	$Stem$	$Syntac$
$Base.$	74.14	74.47	75.53
$n - Lex.$	74.60	75.08	75.58
$n - Head$	74.51	75.32	75.56
$n - Pars.$	75.46	75.90	76.22
Gaz	74.85	74.83	75.92
$Model$	74.83	75.59	75.40
$Comb.$	**76.39**	**76.85**	**77.23**

Table 5: Obtained results when the features were extracted from both hand-aligned and automatically-aligned parallel corpora

provement of using $Comb.$ compared to baseline is statistically significant. We notice again that when the TL baseline MD model uses a richer feature set, the obtained improvement from using RRL becomes smaller. We also observed that automatically aligned data helped capture most of the unseen mentions whereas the hand-aligned features helped decrease the number of false-alarms. It is important to notice that when features $Comb.$ is used with $Stem$ baseline model, the obtained F-measure (76.85) is 1.3 higher than the baseline model which uses lexical, stem and syntactic features – $Syntac$ (75.53). The type of errors which mostly occur and has not been fixed neither by using hand-aligned data, automatically aligned data nor the combination of both are the nominal mentions whose class depends fully on the context. For instance, the word موظف (mwZf - employee) which was considered as **O** by the MD model because it has not been seen in any of the parallel data in a context such as the following:

تعريف شكل الموظف المصري كان ...

transliterated as:

tEryf $kl AlmwZf AlmSry ...

and translated as: *"defining the life of the Egyptian employee ..."*

7 Previous Works

Several research works, in different NLP tasks, have shown that the use of an RRL to achieve a better performance in a resource-challenged language yields to successful results. In (Rogati et al., 2003), authors used a statistical machine translation (MT) system to build an Arabic stemmer. The obtained stemmer has a performance of 87.5%. In (Ide et al., 2002), authors use the aligned versions of George Orwell's *Nineteen Eighty-Four* in seven languages in order to determine sense distinctions which can be used in the Word Sense Disambiguation (WSD) task. They report that the automatically obtained tags are at least as reliable as the one made by human annotators. Similarly, (Ng et al., 2003) report a research study which uses an English-Chinese parallel corpus in order to extract sense-tagged training data. In (Hwa et al., 2002), authors report promising results of inducing Chinese dependency trees from English. The obtained model outperformed the baseline.

One of the significant differences between these works and the one we present in this paper is that instead of using the propagated annotation directly as training data we use it as an additional feature and thus allow the MEMM model to weigh each one of them. By doing so, the model is able to distinguish between the relevant and the irrelevant information propagated from the RRL.

Authors in (Zitouni and Florian, 2008) attempt to enhance an MD model of a foreign language by using an English MD system. They have used an MT system to (i) translate the text to English; (ii) run the English model on the translated text; (iii) and propagate outcome to the original text. The approach in (Zitouni and Florian, 2008) requires a MT system that needs more effort and resources to build when compared to a parallel corpus (used in our experiments); not all institutions may have access to MT and MD systems in plenty of language pairs.

8 Conclusions and Future Works

In this paper, we presented a novel approach that allows to exploit the richness, in terms of resources, of one language (English) to the benefit of a target language (Arabic). We achieved successful results by adopting a novel approach performing in three main steps, namely: (i) Annotate the English side of an English-to-Arabic parallel corpus automatically; (ii) Project the obtained annotation from English to Arabic via the parallel corpus; and (iii) Extract features of different linguistic motivations of the automatically tagged Arabic tokens. Thereafter, each of the extracted features is used to bootstrap Arabic MD system. We use different Arabic baseline MD models which employ different feature sets representing different levels of richness in resources. We also use both a one million word hand-aligned parallel corpus and a 22 million word automatically aligned one in order to study size vs. noise trade-off.

Results show that a statistically significant improvement is always observed even when the Arabic baseline MD model uses all the available resources. When we use the hand-aligned parallel corpus, we obtain up to 2.2 points improvement when the Arabic MD model has access to very limited resources. It decreases to 1.7 points when we use all the resources we could gather for the Arabic language. When no human-annotated data is available in the TL, we show that we can obtain a performance of 57.6 using only mention propagation from RRL. The results also show that a greater improvement is achieved when using a small hand-aligned corpus than using a 20 times bigger automatically aligned data. However, in case both of them are available, combining them leads to even higher results.

References

Yassine Benajiba, Mona Diab, and Paolo Rosso. 2008. Arabic named entity recognition using optimized feature sets. In *Proc. of EMNLP'08*, pages 284–293.

Stanley Chen and Ronald Rosenfeld. 2000. A survey of smoothing techniques for ME models. *IEEE Transaction on Speech and Audio Processing*.

Mona Diab, Kadri Hacioglu, and Dan Jurafsky. 2004. Automatic tagging of arabic text: from raw text to base phrase chunks. In *Proc. of HLT/NAACL'04*.

Joshua Goodman. 2002. Sequential conditional generalized iterative scaling. In *Proceedings of ACL'02*.

Rebecca Hwa, Philip Resnik, and Amy Weinberg. 2002. Breaking the resource bottleneck for multilingual parsing. In *Proceedings of LREC*.

Nancy Ide, Tomaz Erjavec, and Dan Tufis. 2002. Sense discrimination with parallel corpora. In *Proceedings of the SIGLEX/SENSEVAL Workshop on Word Sense Disambiguation*, pages 54–60.

Abe Ittycheriah and Salim Roukos. 2005. A maximum entropy word aligner for arabic-english machine translation. In *Proceedings of HLT/EMNLP'05*, pages 89–96.

Alexandre Klementiev and Dan Roth. 2006. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Proceedings of ACL'06*, pages 817–824, Sydney, Australia. Association for Computational Linguistics.

Young-Suk Lee, Kishore Papineni, Salim Roukos, Ossama Emam, and Hany Hassan. 2003. Language model based Arabic word segmentation. In *Proc. of the ACL'03*, pages 399–406.

Hwee Tou Ng, Bin Wang, and Yee Seng Chan. 2003. Exploiting parallel texts for word sense disambiguation: An empirical study. In *Proceedings of ACL'03*, pages 455–462.

Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses*. John Wiley Sons.

Lance Ramshaw and Mitchell 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*, pages 157–176. Kluwer.

Monica Rogati, Scott McCarley, and Yiming Yang. 2003. Unsupervised learning of arabic stemming using a parallel corpus. In *Proceedings of ACL'03*, pages 391–398.

Eric. F. Tjong Kim Sang. 2002. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2002*, pages 155–158. Taipei, Taiwan.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of HLT'01*, pages 1–8.

Imed Zitouni and Radu Florian. 2008. Mention detection crossing the language barrier. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Honolulu, Hawaii, October.

Imed Zitouni and Radu Florian. 2009. Cross-language information propagation for arabic mention detection. *ACM Transactions on Asian Language Information Processing (TALIP)*, 8(4):1–21.

Imed Zitouni, Jeff Sorensen, Xiaoqiang Luo, and Radu Florian. 2005. The impact of morphological stemming on arabic mention detection and coreference resolution. In *Proc. of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 63–70.

Domain Adaptation of Rule-Based Annotators for Named-Entity Recognition Tasks

Laura Chiticariu Rajasekar Krishnamurthy Yunyao Li Frederick Reiss Shivakumar Vaithyanathan
IBM Research – Almaden
650 Harry Road, San Jose, CA 95120, USA
{chiti, rajase, yunyaoli, frreiss}@us.ibm.com, shiv@almaden.ibm.com

Abstract

Named-entity recognition (NER) is an important task required in a wide variety of applications. While rule-based systems are appealing due to their well-known "explainability," most, if not all, state-of-the-art results for NER tasks are based on machine learning techniques. Motivated by these results, we explore the following natural question in this paper: *Are rule-based systems still a viable approach to named-entity recognition?* Specifically, we have designed and implemented a high-level language *NERL* on top of SystemT, a general-purpose algebraic information extraction system. *NERL* is tuned to the needs of NER tasks and simplifies the process of building, understanding, and customizing complex rule-based named-entity annotators. We show that these customized annotators match or outperform the best published results achieved with machine learning techniques. These results confirm that we can reap the benefits of rule-based extractors' explainability without sacrificing accuracy. We conclude by discussing lessons learned while building and customizing complex rule-based annotators and outlining several research directions towards facilitating rule development.

1 Introduction

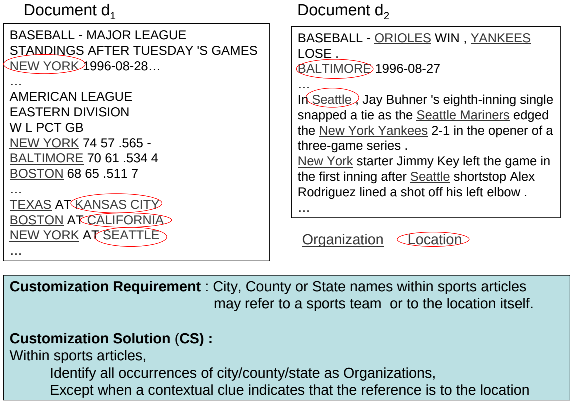
Named-entity recognition (NER) is the task of identifying mentions of rigid designators from text belonging to named-entity types such as persons, organizations and locations (Nadeau and Sekine, 2007). While NER over formal text such as news articles and webpages is a well-studied problem (Bikel et al., 1999; McCallum and Li, 2003; Etzioni et al., 2005), there has been recent work on NER over informal text such as emails and blogs (Huang et al., 2001; Poibeau and Kosseim, 2001; Jansche and Abney, 2002; Minkov et al., 2005; Gruhl et al., 2009). The techniques proposed in the literature fall under three categories: rule-based (Krupka and Hausman, 2001; Sekine and Nobata, 2004), machine learning-based (O. Bender and Ney, 2003; Florian et al., 2003; McCallum and Li, 2003; Finkel and Manning, 2009; Singh et al., 2010) and hybrid solutions (Srihari et al., 2001; Jansche and Abney, 2002).

1.1 Motivation

Although there are well-established rule-based systems to perform NER tasks, most, if not all, state-of-the-art results for NER tasks are based on machine learning techniques. However, the rule-based approach is still extremely appealing due to the associated transparency of the internal system state, which leads to better explainability of errors (Siniakov, 2010). Ideally, one would like to benefit from the transparency and explainability of rule-based techniques, while achieving state-of-the-art accuracy.

A particularly challenging aspect of rule-based NER in practice is domain customization — customizing existing annotators to produce accurate results in new domains. In machine learning-based systems, adapting to a new domain has traditionally involved acquiring additional labeled data and learning a new model from scratch. However, recent work has proposed more sophisticated approaches that learn a domain-independent base model, which can later be adapted to specific domains (Florian et

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1002–1012,
MIT, Massachusetts, USA, 9-11 October 2010.



Document d_1

BASEBALL - MAJOR LEAGUE
STANDINGS AFTER TUESDAY 'S GAMES
NEW YORK 1996-08-28…
…
AMERICAN LEAGUE
EASTERN DIVISION
W L PCT GB
NEW YORK 74 57 .565 -
BALTIMORE 70 61 .534 4
BOSTON 68 65 .511 7
…
TEXAS AT KANSAS CITY
BOSTON AT CALIFORNIA
NEW YORK AT SEATTLE
…

Document d_2

BASEBALL - ORIOLES WIN , YANKEES LOSE .
BALTIMORE 1996-08-27
…
In Seattle , Jay Buhner 's eighth-inning single snapped a tie as the Seattle Mariners edged the New York Yankees 2-1 in the opener of a three-game series .
New York starter Jimmy Key left the game in the first inning after Seattle shortstop Alex Rodriguez lined a shot off his left elbow .
…

Customization Requirement : City, County or State names within sports articles may refer to a sports team or to the location itself.

Customization Solution (CS) :
Within sports articles,
Identify all occurrences of city/county/state as Organizations,
Except when a contextual clue indicates that the reference is to the location

Figure 1: Example Customization Requirement

```
// Core rules identify Organization and Location candidates

// Begin customization
// Identify articles covering sports event from article title
CR1 <SportsArticle> ← Evaluate Regular Expressions <R1>

// Identify locations in sports articles
CR2 Retain <Location> As <LocationMaybeOrg> If ContainedWithin <SportsArticle>

// City/County/State references (e.g., New York) may refer to the sports team in that city
CR3 Retain <LocationMaybeOrg> If Matches Dictionaries
     <'cities.dict','counties.dict','states.dict'>

// Some city references in sports articles may refer to the city (e.g., In Seattle )
// These references should not be reclassified as Organization
CR4 Discard <LocationMaybeOrg> If Matches Regular Expression <R2>
                                    on Left Context 2 Tokens

// City references to sports teams are added to Organization and removed from Location
CR5 Augment <Organization> With <LocationMaybeOrg>
// End customization


// Continuation of core rules
// Remove Locations that overlap with Organizations
     Discard <Location> If Overlaps Concepts <Organization>
```

Figure 2: Example Customization Rules in *NERL*

al., 2004; Blitzer et al., 2006; Jiang and Zhai, 2006; Arnold et al., 2008; Wu et al., 2009). Implementing a similar approach for rule-based NER typically requires a significant amount of manual effort to (a) identify the explicit semantic changes required for the new domain (e.g., differences in entity type definition), (b) identify the portions of the (complex) core annotator that should be modified for each difference and (c) implement the required customization rules without compromising the extraction quality of the core annotator. Domain customization of rule-based NER has not received much attention in the recent literature with a few exceptions (Petasis et al., 2001; Maynard et al., 2003; Zhu et al., 2005).

1.2 Problem Statement

In this paper, we explore the following natural question: Are rule-based systems still a viable approach to named-entity recognition? Specifically, (a) *Is it possible to build, maintain and customize rule-based NER annotators that match the state-of-the-art results obtained using machine-learning techniques?* and (b) *Can this be achieved with a reasonable amount of manual effort?*

1.3 Contributions

In this paper, we address the challenges mentioned above by (i) defining a taxonomy of the different types of customizations that a rule developer may perform when adapting to a new domain (Sec. 2), (ii) identifying a set of high-level operations required for building and customizing NER annotators, and (iii) exposing these operations in a domain-specific NER rule language, *NERL*, developed on top of SystemT (Chiticariu et al., 2010), a general-purpose algebraic information extraction system (Sec. 3). *NERL* is specifically geared towards building and customizing complex NER annotators and makes it easy to understand a complex annotator that may comprise hundreds of rules. It simplifies the identification of what portions need to be modified for a given customization requirement. It also makes individual customizations easier to implement, as illustrated by the following example.

Suppose we have to customize a domain-independent rule-based NER annotator for the CoNLL corpus (Tjong et al., 2003). Consider the two sports-related news articles in Fig. 1 from the corpus, where city names such as '*New York*' or '*Seattle*' can refer to either a *Location* or an *Organization* (the sports team based in that city). In the domain-independent annotator, city names were always identified as *Location*, as this subtle requirement was not considered during rule development. A customization to address this issue is shown in Fig. 1, which can be implemented in *NERL* with five rules (Fig. 2). This customization (explained in detail in Sec. 3) improved the $F_{\beta=1}$ score for *Organization* and *Location* by approximately 9% and 3%, respectively (Sec. 4).

We used *NERL* to customize a domain-independent rule-based NER annotator for three different domains – CoNLL03 (Tjong et al., 2003), Enron (Minkov et al., 2005) and ACE05 (NIST, 2005). Our experimental results (Sec. 4.3) demonstrate that the customized annotators have extraction quality better than the best-known results for

	Affects Single Entity Type	Affects Multiple Entity Types
Identify New Instances	C_S, C_{DD}, C_{DSD}	B_R
Modify Existing instances	C_{EB}, C_{DD}	C_{ATA}, C_G

Table 1: Categorizing NER Customizations

individual domains, which were achieved with machine learning techniques. The fact that we are able to achieve such results across multiple domains answers our earlier question and confirms that we can reap the benefits of rule-based extractors' explainability without sacrificing accuracy.

However, we found that even using *NERL*, the amount of manual effort and expertise required in rule-based NER may still be significant. In Sec. 5, we report on the lessons learned and outline several interesting research directions towards simplifying rule development and facilitating the adoption of the rule-based approach towards NER.

2 Domain Customization for NER

We consider NER tasks following the broad definition put forth by (Nadeau and Sekine, 2007), formally defined as follows:

Definition 1 *Named entity recognition is the task of identifying and classifying mentions of entities with one or more rigid designators, as defined by (Kripke, 1982).*

For instance, the identification of proper nouns representing persons, organizations, locations, product names, proteins, drugs and chemicals are considered as NER tasks.

Based on our experience of customizing NER annotators for multiple domains, we categorize the customizations involved into two main categories as listed below. This categorization motivates the design of *NERL* (Sec. 3).

Data-driven (C_{DD}): The most common NER customization is data-driven, where the customizations mostly involve the addition of new patterns and dictionary entries, driven by observations from the training data in the new domain. An example is the addition of a new rule to identify locations from the beginning of news articles (e.g., "BALTIMORE 1995-08-27" and "MURCIA , Spain 1996-09-10").

Application-driven: What is considered a valid named entity and its corresponding type can vary across application domains. The most common dimensions on which the definition of a named entity can vary are:

Entity Boundary (C_{EB}): Different application domains may have different definitions of where the same entity starts or ends. For example, a *Person* may (CoNLL03) or may not (Enron) include generational markers (e.g. "Jr." in "Bush Jr." or "IV" in "Henry IV").

Ambiguous Type Assignment (C_{ATA}): The exact type of a given named entity can be ambiguous. Different applications may assign different types for the same named entity. For instance, all instances of "White House" may be considered as *Location* (CoNLL03), or be assigned as *Facility* or *Organization* based on their context (ACE05). In fact, even within the same application domain, entities typically considered as of the same type may be assigned differently. For example, given "New York beat Seattle" and "Ethiopia beat Uganda", both '*New York*' and '*Ethiopia*' are teams referred by their locations. However, (Tjong et al., 2003) considers the former, which corresponds to a city, as an *Organization*, and the latter, which corresponds to a country, as a *Location*.

Domain-Specific Definition (C_{DSD}): Whether a given term is even considered a named entity may depend on the specific domain. As an example, consider the text "Commercialization Meeting - SBeck, BHall, BSuperty, TBusby, SGandhi-Gupta". Informal names such as '*SBeck*' and '*BHall*' may be considered as valid person names (Enron).

Scope(C_S): Each type of named entity usually contains several subtypes. For the same named entity task, different applications may choose to include different sets of subtypes. For instance, roads and buildings are considered part of *Location* in CoNLL03, while they are not included in ACE05.

Granularity(C_G): Name entity types are hierarchical. Different applications may define NER tasks at different granularities. For instance, in ACE05, *Organization* and *Location* entity types were split into four entity types (*Organization*, *Location*, *Geo-Political Entity* and *Facility*).

The different customizations are summarized as shown in Tab. 1, based on the following criteria: (i) whether the customization identifies new instances or modifies existing instances; and (ii) whether the customization affects single or multiple entities. For instance, C_S identifies new instances for a single entity type, as it adds instances of a new subtype for an existing entity type. Note that B_R in the table denotes the rules used to build the core annotator.

3 Named Entity Rule Language

3.1 Grammar vs. Algebraic NER

Traditionally, rule-based NER systems were based on the popular CPSL cascading grammar specification (Appelt and Onyshkevych, 1998). CPSL is designed so that rules that adhere to the standard can be executed efficiently with finite state transducers. Accordingly, the standard defines a rigid left-to-right execution model where a region of text can be matched by at most one rule according to a fixed rule priority, and where overlapping annotations are disallowed in the output of each grammar phase.

While it simplifies the design of CPSL engines, the rigidity of the rule matching semantics makes it difficult to express operations frequently used in rule-based information extraction. These limitations have been recognized in the literature, and several extensions have been proposed to allow more flexible matching semantics, and to allow overlapping annotations (Cunningham et al., 2000; Boguraev, 2003; Drozdzynski et al., 2004). However, even with these extensions, common operations such as filtering annotations (e.g. CR_4 in Fig. 2), are difficult to express in grammars and often require an escape to custom procedural code.

Recently, several declarative algebraic languages have been proposed for rule-based IE systems, notably AQL (Chiticariu et al., 2010) and Xlog (Shen et al., 2007). These languages are not constrained by the requirement that all rules map onto finite state transducers, and therefore can express a significantly richer semantics than grammar-based languages. In particular, the AQL rule language as implemented in SystemT (Chiticariu et al., 2010) can express many common operations used in rule-based information extraction without requiring custom code. In addition, the separation of extraction semantics from execution enables SystemT's rule optimizer and efficient runtime engine. Indeed, as shown in (Chiticariu et al., 2010), SystemT can deliver an order of magitude higher annotation throughput compared to a state-of-the-art CPSL-based IE system.

Since AQL is a general purpose information extraction rule language, similar to CPSL and JAPE, it exposes an expressive set of capabilities that go beyond what is required for NER tasks. These additional capabilities can make AQL rules more verbose than is necessary for implementing rules in the NER domain. For example, Fig. 3 shows how the same customization rule CR_4 from Fig. 2 can be implemented in JAPE or in AQL. Notice how implementing even a single customization may lead to defining complex rules (e.g. JAPE-R_1, AQL-R_1) and sometimes even using custom code (e.g. JAPE-R_2). As illustrated by this example, the rules in AQL and JAPE tend to be complex since some operations — e.g., filtering the outputs of one rule based on the outputs of another rule — that are common in NER rule sets require multiple rules in AQL or multiple grammar phases in JAPE.

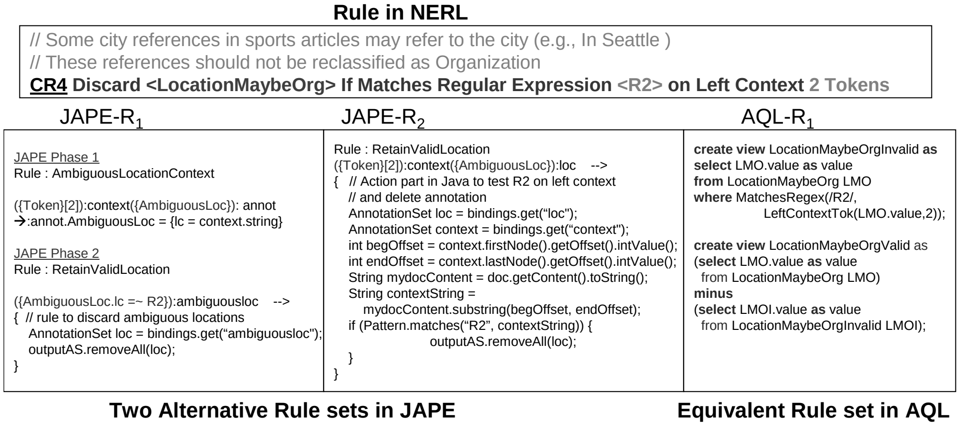
To make NER rules easier to develop and to understand, we designed and implemented Named Entity Rule Language (*NERL*) on top of SystemT. *NERL* is a declarative rule language designed specifically for named entity recognition. The design of *NERL* draws on our experience with building and customizing multiple complex NER annotators. In particular, we have identified the operations required in practice for such tasks, and expose these operations as built-in constructs in *NERL*. In doing so, we ensure that frequently performed operations can be expressed succinctly, so as not to complicate the rule set unnecessarily. As a result, *NERL* rules for named entity recognition tasks are significantly more compact and easy to understand than the equivalent AQL rules. At the same time, *NERL* rules can easily be compiled to AQL, allowing our NER rule development framework to take advantage of the capabilities of the SystemT rule optimizer and efficient runtime execution engine.

3.2 *NERL*

For the rest of this section, we focus on describing the types of rules supported in *NERL*. In Sec. 4, we shall demonstrate empirically that *NERL* can be successfully employed in building and customizing complex NER annotators.

A *NERL* rule has the following form:

$$\textit{IntConcept} \leftarrow RuleBody(\textit{IntConcept}_1, \textit{IntConcept}_2, \ldots)$$



Rule in NERL

```
// Some city references in sports articles may refer to the city (e.g., In Seattle )
// These references should not be reclassified as Organization
CR4 Discard <LocationMaybeOrg> If Matches Regular Expression <R2> on Left Context 2 Tokens
```

JAPE-R$_1$

```
JAPE Phase 1
Rule : AmbiguousLocationContext

({Token}[2]):context({AmbiguousLoc}): annot
→:annot.AmbiguousLoc = {lc = context.string}

JAPE Phase 2
Rule : RetainValidLocation

({AmbiguousLoc.lc =~ R2}):ambiguousloc    -->
{  // rule to discard ambiguous locations
   AnnotationSet loc = bindings.get(“ambiguousloc");
   outputAS.removeAll(loc);
}
```

JAPE-R$_2$

```
Rule : RetainValidLocation
({Token}[2]):context({AmbiguousLoc}):loc    -->
{   // Action part in Java to test R2 on left context
    // and delete annotation
    AnnotationSet loc = bindings.get(“loc");
    AnnotationSet context = bindings.get(“context");
    int begOffset = context.firstNode().getOffset().intValue();
    int endOffset = context.lastNode().getOffset().intValue();
    String mydocContent = doc.getContent().toString();
    String contextString =
       mydocContent.substring(begOffset, endOffset);
    if (Pattern.matches(“R2”, contextString)) {
                      outputAS.removeAll(loc);
    }
}
```

AQL-R$_1$

```
create view LocationMaybeOrgInvalid as
select LMO.value as value
from LocationMaybeOrg LMO
where MatchesRegex(/R2/,
              LeftContextTok(LMO.value,2));

create view LocationMaybeOrgValid as
(select LMO.value as value
  from LocationMaybeOrg LMO)
minus
(select LMOI.value as value
  from LocationMaybeOrgInvalid LMOI);
```

Two Alternative Rule sets in JAPE

Equivalent Rule set in AQL

Figure 3: Single Customization Rule expressed in *NERL*, JAPE and AQL

Intuitively, a *NERL* rule creates an intermediate concept or named entity (*IntConcept* for short) by applying a *NERL* rule on the input text and zero or more previously defined intermediate concepts.

***NERL* Rule Types** The types of rules supported in *NERL* are summarized in Tab. 2. In what follows, we illustrate these types by means of examples.

Feature definition (FD): FD rules identify basic features from text (e.g., *FirstName*, *LastName* and *CapsWord* features for identifying person names).

Candidate definition (CD): CD rules identify complete occurrences of the target entity. For instance, the Sequence rule "*LastName followed by ',' followed by FirstName*" identifies person annotations as a sequence of three tokens, where the first and third tokens occur in dictionaries containing last and first names.

Candidate Refinement (CR): CR rules are used to refine candidates generated for different annotation types. E.g., the *Filter* rule $CR3$ in Fig. 2 retains *LocationMaybeOrg* annotations that appear in one of several dictionaries.

Consolidation (CO): CO rules are used to resolve overlapping candidates generated by multiple CD rules. For instance, consider the text "Please see the following request from Dr. Kenneth Lim of the BAAQMD.". A CD rule may identify '*Dr. Kenneth Lim*' as a person, while another CD rule may identify '*Kenneth Lim*' as a candidate person. A consolidation rule is then used to merge these two annotations to produce a single annotation for '*Dr. Kenneth Lim*'.

***NERL* Examples** Within these categories, three types of rules deserve special attention, as they correspond to frequently used operations and are specifically designed to ensure compactness of the ruleset. In contrast, as discussed earlier (Fig. 3), each of these operations require several rules and possibly custom code in existing rule-based IE systems.

DynamicDict: The *DynamicDict* rule is used to create customized gazetteers on the fly. The following example shows the need for such a rule: While '*Clinton*' does not always refer to a person's last name (Clinton is the name of several cities in USA), in documents containing a full person name with '*Clinton*' as a last name (e.g., '*Hillary Clinton*') it is reasonable to annotate all references to the (possibly) ambiguous word '*Clinton*' as a person. This goal can be accomplished using the rule <*Create Dynamic Dictionary using Person with length 1 to 2 tokens*>, which creates a gazetteer on a per-document basis.

Filter: The *Filter* rule is used to discard/retain certain intermediate annotations based on predicates on the annotation text and its local context. Example filtering predicates include

- Discard C If Matches Regular Expression R
- Retain C If Contains Dictionary D on Local Context LC
- Discard C If Overlaps Concepts $C_1, C_2, \ldots$

ModifySpan: The *ModifySpan* rule is used to expand or trim the span of a candidate annotation. For instance, an *Entity Boundary* customization to include generational markers as part of a *Person* annotation can be implemented using a *ModifySpan* rule <*Expand Person Using Dictionary 'generation.dict' on RightContext 2 Tokens*>.

Using *NERL* Tab. 2 shows how different types of rules are used during rule building and customizations. Since B_R and C_S involve identifying one

Rule	*Category*	*Syntax*	B_R	C_{DD}		C_G
			C_S	C_{DSD}	C_{EB}	C_{ATA}
Dictionary	FD	**Evaluate Dictionaries** $< D_1, D_2, \ldots >$ with flags?	X	X		
Regex	FD	**Evaluate Regular Expressions** $< R_1, R_2, \ldots >$ with flags?	X	X		
PoS	FD	**Evaluate Part of Speech** $< P_1, P_2, \ldots >$ with language $< L >$?	X	X		
DynamicDict	FD	**Create Dynamic Dictionary** using *IntConcept* with flags?	X	X		
Sequence	CD	*IntConceptorString* multiplicity? (**followed by** *IntConceptorString* multiplicity?)+	X	X		
Filter	CR	**Discard/Retain** *IntConcept*(As *IntConcept*)? **If SatisfiesPredicate** on LocalContext	X	X		X
ModifySpan	CR	**Trim/Expand** *IntConcept* Using Dictionary $< D >$ on LocalContext	X		X	
Augment	CO	**Augment** *IntConcept* With *IntConcept*	X			X
Consolidate	CO	**Consolidate** *IntConcept* using ConsolidationPolicy	X			X

Table 2: Description of rules supported in *NERL*

or more entity (sub)types from scratch, all types of rules are used. C_{DD} and C_{DSD} identify additional instances for an existing type and therefore mainly rely on FD and CD rules. On the other hand, the customizations that modify existing instances (C_{EB},C_{ATA},C_G) require CR and CO rules.

Revisiting the example in Fig. 2, CR rules were used to implement a fairly sophisticated customization in a compact fashion, as follows. Rule CR_1 first identifies sports articles using a regular expression based on the article title. Rule CR_2 marks Locations within these articles as *LocationMaybeOrg* and Rule CR_3 only retains those occurrences that match a city, county or state name (e.g., '*Seattle*'). Rule CR_4 identifies occurrences that have a contextual clue confirming that the mention was to a location (e.g., '*In*' or '*At*'). These occurrences are already classified correctly as *Location* and do not need to be changed. Finally, CR_5 adds the remaining ambiguous mentions to *Organization*, which would be deleted from *Location* by a subsequent core rule.

4 Development and Customization of NER extractors with *NERL*

Using *NERL*, we have developed CoreNER, a domain-independent generic library for multiple NER extraction tasks commonly encountered in practice, including *Person*, *Organization*, *Location*, *EmailAddress*, *PhoneNumber*, *URL*, and *DateTime*, but we shall focus the discussion on the first three tasks (see Tab. 3 for entity definitions), since they are the most challenging. In this section, we first overview the process of developing CoreNER (Sec. 4.1). We then describe how we have customized CoreNER for three different domains (Sec. 4.2), and present a quality comparison with best published results obtained with state-of-the-art machine learning techniques (Sec. 4.3). The tasks we consider are not restricted to documents in a particular language, but due to limited availability of non-English corpora and extractors for comparison, our evaluation uses English-language text. In Sec. 5 we shall elaborate on the difficulties encountered while building and customizing CoreNER using *NERL* and the lessons we learned in the process.

4.1 Developing CoreNER

We have built our domain independent CoreNER library using a variety of formal and informal text (e.g. web pages, emails, blogs, etc.), and information from public data sources such as the US Census Bureau (Census, 2007) and Wikipedia.

The development process proceeded as follows. We first collected dictionaries for each entity type from different resources, followed by manual cleanup when needed to categorize entries collected into "strong" and "weak" dictionaries. For instance, we used US Census data to create several name dictionaries, placing ambiguous entries such as '*White*' and '*Price*' in a dictionary of *ambiguous last names*, while unambiguous entries such as '*Johnson*' and '*Williams*' went to the dictionary for *strict last names*. Second, we developed FD and CD rules to identify candidate entities based on the way named entities generally occur in text. E.g., <*Salutation CapsWord CapsWord*> and <*FirstName*

Type	Subtypes
PER	individual
LOC	Address, Boundary, Land-Region-Natural, Region-General, Region-International, Airport, Buildings-Grounds, Path, Plant, Subarea-Facility, Continent, Country-or-District, Nation, Population-Center, State-or-Province
ORG	Commercial, Educational, Government, Media, Medical-Science Non-Governmental

Table 3: NER Task Types and Subtypes

LastName> for *Person*, and <*CapsWord*{*1,3*} *OrgSuffix*> and <*CapsWord*{*1,2*} *Industry*> for *Organization*. We then added CR and CO rules to account for contextual clues and overlapping annotations (e.g., Delete *Person* annotations appearing within an *Organization* annotation).

The final CoreNER library consists of 104 FD (involving 68 dictionaries, 33 regexes and 3 dynamic dictionaries), 74 CD, 123 CR and 102 CO rules.

4.2 Customizing CoreNER

In this section we describe the process of customizing our domain-independent CoreNER library for several different datasets. We start by discussing our choice of datasets to use for customization.

Datasets For a rigorous evaluation of CoreNER's customizability, we require multiple datasets satisfying the following criteria: First, the datasets must cover diverse sources and styles of text. Second, the set of the most challenging NER tasks *Person*, *Organization* and *Location* (see Tab. 3) considered in CoreNER should be applicable to them. Finally, they should be publicly available and preferably have associated published results, against which we can compare our experimental results. Towards this end, we chose the following public datasets.

- CoNLL03 (Tjong et al., 2003): a collection of Reuters news stories. Consists of formal text.
- Enron (Minkov et al., 2005): a collection of emails with meeting information from the Enron dataset. Contains predominantly informal text.
- ACE05 (NIST, 2005)[1] a collection of broadcast news, broadcast conversations and newswire reports. Consists of both formal and informal text.

Customization Process The goal of customization is to refine the original CoreNER (hence referred to as CoreNER$_{orig}$) in order to improve its extraction quality on the training set (in terms of F$_{\beta=1}$) for each dataset individually. In addition, a development set is available for CoNLL03 (referred to as CoNLL03$_{dev}$), therefore we seek to improve F$_{\beta=1}$ on CoNLL03$_{dev}$ as well.

The customization process for each dataset proceeded as follows. First, we studied the entity definitions and identified their differences when compared with the definitions used for CoreNER$_{orig}$ (Tab. 3). We then added rules to account for the differences. For example, the definition of *Organization* in the CoNLL03 dataset contained a sports organization subtype, which was not considered when developing CoreNER. Therefore, we have used public data sources (e.g., Wikipedia) to collect and curate dictionaries of major sports associations and sport clubs from around the world. The new dictionaries, along with regular expressions identifying sports teams in sports articles were used for defining FD and CD rules such as CR_1 (Fig. 2). Finally, CR and CO rules were added to filter invalid candidates and augment the *Organization* type with the new sports subtype (similar in spirit to rules CR_4 and CR_5 in Fig. 2).

In addition to the train and development sets, the customization process for CoNLL03 also involved unlabeled data from the corpus as follows. 1) Since data-driven rules (C_{DD}) are often created based on a few instances from the training data, testing them on the unlabeled data helped fine tune the rules for precision. 2) CoNLL03 is largely dominated by sports news, but only a subset of all sports were represented in the train dataset. Using the unlabeled data, we were able to add C_{DD} rules for five additional types of sports, resulting in 0.31% improvement in F$_{\beta=1}$ score on CoNLL03$_{dev}$. 3) Unlabeled data was also useful in identifying domain-specific gazetteers by using simple extraction rules followed by a manual cleanup phase. For instance, for CoNLL03 we collected five gazetteers of organization and person names from the unlabeled data, resulting in 0.45% improvement in recall for CoNLL03$_{dev}$.

The quality of the customization on the train collections is shown in Tab. 5. The total number of rules added during customization for each of the three domains is listed in Tab. 4. Notice how rules of all four types are used both in the development

[1] The evaluation test set is not publicly available. Thus, following the example of (Florian et al., 2006), the publicly available set is split into a 80%/20% data split, with the last 20% of the data in chronological order selected as test data.

	FD	CD	CR	CO
CoreNER$_{orig}$	104	74	123	102
CoreNER$_{conll}$	179	56	284	71
CoreNER$_{enron}$	13	10	9	1
CoreNER$_{ace}$	83	35	117	26

Table 4: Rules added during customization

	Precision	**Recall**	**F$_{\beta=1}$**
CoreNER$_{conll}$	97.64	95.60	96.61
CoreNER$_{enron}$	91.15	92.58	91.86
CoreNER$_{ace}$	92.32	91.22	91.77

Table 5: Quality of customization on train datasets (%)

of the domain independent NER annotator, and during customizations for different domains. A total of 8 person weeks were spent on customizations, and we believe this effort is quite reasonable by rule-based extraction standards. For example, (Maynard et al., 2003) reports that customizing the ANNIE domain independent NER annotator developed using the JAPE grammar-based rule language for the ACE05 dataset required 6 weeks (and subsequent tuning over the next 6 months), resulting in improving the quality to 82% for this dataset. As we shall discuss shortly, *with similar manual effort, we were able to achieve results outperforming state-of-art published results on three different datasets, including* ACE05. However, one may rightfully argue that the process is still too lengthy impeding the widespread deployment of rule-based NER extraction. We elaborate on the effort involved and the lessons learned in the process in Sec. 5.

4.3 Evaluation of Customization

We now present an experimental evaluation of the customizability of CoreNER. The main goals are to investigate: (i) the feasibility of CoreNER customization for different application domains; (ii) the effectiveness of such customization compared to state-of-the-art results; (iii) the impact of different types of customization (Tab. 1); and (iv) how often different categories of *NERL* rules (Tab. 2) are used during customization.

We measured the effectiveness of customization using the improvement in extraction quality of the customized CoreNER over CoreNER$_{orig}$. As shown in Tab. 6, customization significantly improved F$_{\beta=1}$ score for CoreNER$_{orig}$ across all datasets. [2]

		Precision	**Recall**	**F$_{\beta=1}$**
CoNLL03$_{dev}$	CoreNER$_{orig}$	83.81	61.77	71.12
	CoreNER$_{conll}$	96.49	93.76	95.11
	Improvement	12.68	31.99	13.99
CoNLL03$_{test}$	CoreNER$_{orig}$	77.21	54.87	64.15
	CoreNER$_{conll}$	93.89	89.75	91.77
	Improvement	15.68	34.88	27.62
Enron	CoreNER$_{orig}$	85.06	69.55	76.53
	CoreNER$_{enron}$	88.41	82.39	85.29
	Improvement	3.35	12.84	8.76
ACE2005	CoreNER$_{orig}$	57.23	57.41	57.32
	CoreNER$_{ace}$	90.11	87.82	88.95
	Improvement	32.88	30.41	31.63

Table 6: Overall Improvement due to Customization (%)

			Precision	**Recall**	**F$_{\beta=1}$**
CoNLL03$_{dev}$	LOC	CoreNER$_{conll}$	**97.17**	95.37	**96.26**
		Florian	96.59	**95.65**	96.12
	ORG	CoreNER$_{conll}$	**93.70**	88.67	**91.11**
		Florian	90.85	**89.63**	90.24
	PER	CoreNER$_{conll}$	**97.79**	95.87	**96.82**
		Florian	96.08	**97.12**	96.60
CoNLL03$_{test}$	LOC	CoreNER$_{conll}$	**93.11**	91.61	**92.35**
		Florian	90.59	**91.73**	91.15
	ORG	CoreNER$_{conll}$	**92.25**	**85.31**	**88.65**
		Florian	85.93	83.44	84.67
	PER	CoreNER$_{conll}$	**96.32**	92.39	**94.32**
		Florian	92.49	**95.24**	93.85
Enron	PER	CoreNER$_{enron}$	**87.27**	**81.82**	**84.46**
		Minkov	81.1	74.9	77.9

Table 7: Comparison with state-of-the-art results(%)

We note that the extraction quality of CoreNER$_{orig}$ was low on CoNLL03 and ACE05 mainly due to differences in entity type definitions. In particular, sports organizations, which occurred frequently in the CoNLL03 collection, were not considered during the development of CoreNER$_{orig}$, while in ACE05, *ORG* and *LOC* entity types were split into four entity types (*Organization*, *Location*, *Geo-Political Entity* and *Facility*). Customizations such as C_S and C_G address the above changes in named-entity type definition and substantially improve the extraction quality of CoreNER$_{orig}$.

Next, we compare the extraction quality of the

[2] CoNLL03$_{dev}$ and CoNLL03$_{test}$ correspond to the development and test sets for CoNLL03 respectively.

customized CoreNER for CoNLL03 and Enron[3] with the corresponding best published results by (Florian et al., 2003) and (Minkov et al., 2005). Tab. 7 shows that our customized CoreNER *outperforms the corresponding state-of-the-art numbers for all the NER tasks* on both CoNLL03 and Enron. [4] These results demonstrate that high-quality annotators can be built by customizing CoreNER$_{orig}$ using *NERL*, with the final extraction quality matching that of state-of-the-art machine learning-based extractors.

It is worthwhile noting that the best published results for CoNLL03 (Florian et al., 2003) were obtained by using four different classifiers (Robust Risk Minimization, Maximum Entropy, Transformation-based learning, and Hidden Markov Model) and trying six different classifier combination methods. Compared to the best published result obtained by combining the four classifiers, the individual classifiers performed between 2.5-7.6% worse for *Location*, 5.6-15.2% for *Organization* and 3.9-14.0% for *Person*[5]. Taking this into account, the extraction quality advantage of customized CoreNER is significant when compared with the individual state-of-the-art classifiers.

Impact of Customizations by Type. While customizing CoreNER for the three datasets, all types of changes described in Sec. 2 were performed. We measured the impact of each type of customization by comparing the extraction quality of CoreNER$_{orig}$ with CoreNER$_{orig}$ enhanced with all the customizations of that type. From the results for CoNLL03 (Tab. 8), we make the following observations.

	# rules added			**Precision**	**Recall**	$\mathbf{F}_{\beta=1}$
C_{EB}	3	CoNLL03$_{dev}$	LOC	↑0.21	↑0.22	↑0.22
			ORG	↑1.35	↑0.38	↑0.59
			PER	-	-	-
		CoNLL03$_{test}$	LOC	↑0.30	↑0.36	↑0.33
			ORG	↑0.54	↑0.12	↑0.20
			PER	-	-	-
C_{ATA}	5	CoNLL03$_{dev}$	LOC	↑7.18	↓0.87	↑3.19
			ORG	↑1.37	↑10.67	↑9.04
			PER	↓0.04	-	↓0.01
		CoNLL03$_{test}$	LOC	↑7.73	↓1.20	↑3.77
			ORG	↑1.37	↑11.62	↑14.18
			PER	-	-	-
C_{DSD}	2	CoNLL03$_{dev}$	LOC	↑0.85	-	↑0.45
			ORG	↑1.00	↓0.07	↑0.01
			PER	-	-	-
		CoNLL03$_{test}$	LOC	↑0.04	↓0.12	↓0.12
			ORG	↑0.64	-	↑0.04
			PER	-	-	-
C_S	149	CoNLL03$_{dev}$	LOC	↑1.63	↓0.21	↑0.85
			ORG	↑11.44	↑40.79	↑39.73
			PER	↑0.13	-	↑0.05
		CoNLL03$_{test}$	LOC	↑3.71	↓0.18	↑2.05
			ORG	↑9.2	↑36.24	↑37.96
			PER	↑0.58	-	↑0.2
C_{DD}	431	CoNLL03$_{dev}$	LOC	↓0.94	↑10.18	↑3.99
			ORG	↑9.63	↑11.93	↑14.71
			PER	↑6.12	↑28.5	↑18.84
		CoNLL03$_{test}$	LOC	↓1.66	↑6.72	↑1.64
			ORG	↑8.84	↑12.40	↑15.90
			PER	↑9.15	↑31.48	↑22.21

Table 8: Impact by customization type on CoNLL03(%)

- Customizations that identify additional subtypes of entities (C_S) or modify existing instances for multiple types (C_{ATA}) have significant impact. This effect can be especially high when the missing subtype appears very often in the new domain (E.g., over 50% of the organizations in CoNLL03 are sports teams).
- Data-driven customizations (C_{DD}) rely on the aggregated impact of many rules. While individual rules may have considerable impact on their own (e.g., identifying all names that appear as part of a player list increases the recall of PER by over 6% on both CoNLL03$_{dev}$ and CoNLL03$_{test}$), the overall impact relies on the accumulative effect of many small improvements.
- Certain customizations (C_{EB} and C_{DSD}) provide smaller quality improvements, both per rule and in aggregate.

5 Lessons Learned

Our experimental evaluation shows that rule-based annotators can achieve quality comparable to that of state-of-the-art machine learning techniques. In this section we discuss three important lessons learned regarding the human effort involved in developing such rule-based extractors.

Usefulness of *NERL* We found *NERL* very helpful in that it provided a higher-level abstraction catered specifically towards NER tasks, thus hiding the complexity inherent in a general-purpose IE rule language. In doing so, *NERL* restricts the large space of operations possible within a general-purpose language to the small number of predefined "templates"

[3] We cannot meaningfully compare our results against previously published results for ACE05, which is originally used for mention detection while CoreNER considers only NER tasks.

[4] For Enron the comparison is reported only for *Person*, as labeled data is available only for that type.

[5] Extended version obtained via private communication.

listed in Tab. 2. (We have shown empirically that our choice of *NERL* rules is sufficient to achieve high accuracy for NER tasks.) Therefore, *NERL* simplifies development and maintenance of complex NER extractors, since one does not need to worry about multiple AQL statements or JAPE grammar phases for implementing a single conceptual operation such as filtering (see Fig. 3).

Is *NERL* Sufficient? Even using *NERL*, building and customizing NER rules remains a labor intensive process. Consider the example of designing the filter rule CR_4 from Fig. 3. First, one must examine multiple false positive *Location* entities to even decide that a filter rule is appropriate. Second, one must understand how those false positives were produced, and decide accordingly on the particular concept to be used as filter (*LocationMaybeOrg* in this case). Finally, one needs to decide how to build the filter. Tab. 9 lists all the attributes that need to be specified for a *Filter* rule, along with examples of the search space for each rule attribute.

Rule Attributes	**Examples of Search Space**
Location	Intermediate Concept to filter
Predicate Type	Matches Regex, Contains Dictionary, . . .
Predicate Parameter	Regular Expressions, Dictionary Entries, . . .
Context Type	Entity text, Left or Right context
Context Parameter	k tokens, l characters

Table 9: Search space explored while adding a *Filter* rule

This search space problem is not unique to filter rules. In fact, most rules in Tab. 2 have two or more rule attributes. Therefore, designing an individual *NERL* rule remains a time-consuming "trial and error" process, in which multiple "promising" combinations are implemented and evaluated individually before deciding on a satisfactory final rule.

Tooling for *NERL* The fact that *NERL* is a high-level language exposing a restricted set of operators can be exploited to reduce the human effort involved in building NER annotators by enabling the following tools:

Annotation Provenance Tools tracking provenance (Cheney et al., 2009) for *NERL* rules can help in explaining exactly which sequence of rules is responsible for producing a given false positive, thereby enabling one to quickly identify "misbehaved" rules. For instance, one can quickly narrow down the choices for the location where the filter rule CR_4 (Fig. 2) should be applied based on the provenance of the false positives. Similarly, tools for explaining false positives in the spirit of (Huang et al., 2008), are also conceivable.

Automatic Parameter Learning The most time-consuming part in building a rule often is to decide the value of its parameters, especially for FD and CR rules. For instance, while defining a CR rule, one has to choose values for the Predicate parameter and the Context parameter (see Tab. 9). Some parameter values can be learned – for example, dictionaries (Riloff, 1993) and regular expressions (Li et al., 2008).

Automatic Rule Refinement Tools automatically suggesting entire customization rules to a complex *NERL* program in the spirit of (Liu et al., 2010) can further reduce human effort in building NER annotators. With the help of such tools, one only needs to consider good candidate *NERL* rules suggested by the system without having to go through the conventional manual "trial and error" process.

6 Conclusion

In this paper, we described *NERL*, a high-level rule language for building and customizing NER annotators. We demonstrated that a complex NER annotator built using *NERL* can be effectively customized for different domains, achieving extraction quality superior to the state-of-the-art numbers. However, our experience also indicates that the process of designing the rules themselves is still manual and time-consuming. Finally, we discuss how *NERL* opens up several interesting research directions towards the development of sophisticated tooling for automating some of the rule development tasks.

References

D. E. Appelt and B. Onyshkevych. 1998. The common pattern specification language. In *TIPSTER workshop*.

A. Arnold, R. Nallapati, and W. W. Cohen. 2008. Exploiting feature hierarchy for transfer learning in named entity recognition. In *ACL*.

D. M. Bikel, R. L. Schwartz, and R. M. Weischedel. 1999. An algorithm that learns what's in a name. In *Machine Learning*, volume 34, pages 211–231.

J. Blitzer, R. Mcdonald, and F. Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*.

B. Boguraev. 2003. Annotation-based finite state processing in a large-scale nlp arhitecture. In *RANLP*.

Census. 2007. U.S. Census Bureau. http://www.census.gov.

J. Cheney, L. Chiticariu, and W. Tan. 2009. Provenance in databases: Why, how, and where. *Foundations and Trends in Databases*, 1(4):379–474.

L. Chiticariu, R. Krishnamurthy, Y. Li, S. Raghavan, F. Reiss, and S. Vaithyanathan. 2010. SystemT: An algebraic approach to declarative information extraction. In *ACL*.

H. Cunningham, D. Maynard, and V. Tablan. 2000. JAPE: a Java Annotation Patterns Engine (Second Edition). Research Memorandum CS–00–10, Department of Computer Science, University of Sheffield.

W. Drozdzynski, H. Krieger, J. Piskorski, U. Schäfer, and F. Xu. 2004. Shallow processing with unification and typed feature structures — foundations and applications. *Künstliche Intelligenz*, 1:17–23.

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. *Artif. Intell.*, 165(1):91–134.

J. R. Finkel and C. D. Manning. 2009. Nested named entity recognition. In *EMNLP*.

R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. 2003. Named entity recognition through classifier combination. In *CoNLL*.

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 *NAACL-HLT*.

R. Florian, H. Jing, N. Kambhatla, and I. Zitouni. 2006. Factorizing complex models: A case study in mention detection. In *ACL*.

D. Gruhl, M. Nagarajan, J. Pieper, C. Robson, and A. Sheth. 2009. Context and domain knowledge enhanced entity spotting in informal text. In *ISWC*.

J. Huang, G. Zweig, and M. Padmanabhan. 2001. Information extraction from voicemail. In *ACL*.

Jiansheng Huang, Ting Chen, AnHai Doan, and Jeffrey F. Naughton. 2008. On the Provenance of Non-Answers to Queries over Extracted Data. *PVLDB*, 1(1):736–747.

M. Jansche and S. Abney. 2002. Information extraction from voicemail transcripts. In *EMNLP*.

J. Jiang and C. Zhai. 2006. Exploiting domain structure for named entity recognition. In *NAACL-HLT*.

Saul Kripke. 1982. *Naming and Necessity*. Harvard University Press.

G. R. Krupka and K. Hausman. 2001. IsoQuest Inc.: Description of the NetOwlTM extractor system as used for MUC-7. In *MUC-7*.

Y. Li, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, and H. V. Jagadish. 2008. Regular expression learning for information extraction. In *EMNLP*.

B. Liu, L. Chiticariu, V. Chu, H. V. Jagadish, and F. Reiss. 2010. Automatic Rule Refinement for Information Extraction. *PVLDB*, 3.

D. Maynard, K. Bontcheva, and H. Cunningham. 2003. Towards a semantic extraction of named entities. In *RANLP*.

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

E. Minkov, R. C. Wang, and W. W. Cohen. 2005. Extracting personal names from emails: Applying named entity recognition to informal text. In *HLT/EMNLP*.

D. Nadeau and S. Sekine. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26.

NIST. 2005. The ace evaluation plan.

F. J. Och O. Bender and H. Ney. 2003. Maximum entropy models for named entity recognition. In *CoNLL*.

G. Petasis, F. Vichot, F. Wolinski, G. Paliouras, V. Karkaletsis, and C. Spyropoulos. 2001. Using machine learning to maintain rule-based named-entity recognition and classification systems. In *ACL*.

T. Poibeau and L. Kosseim. 2001. Proper name extraction from non-journalistic texts. In *Computational Linguistics in the Netherlands*, pages 144–157.

E. Riloff. 1993. Automatically constructing a dictionary for information extraction tasks. In *KDD*.

S. Sekine and C. Nobata. 2004. Definition, dictionaries and tagger for extended named entity hierarchy. In *Conference on Language Resources and Evaluation*.

W. Shen, A. Doan, J. F. Naughton, and R. Ramakrishnan. 2007. Declarative information extraction using datalog with embedded extraction predicates. In *VLDB*.

S. Singh, D. Hillard, and C. Leggeteer. 2010. Minimally-supervised extraction of entities from text advertisements. In *NAACL-HLT*.

P. Siniakov. 2010. *GROPUS - an adaptive rule-based algorithm for information extraction*. Ph.D. thesis, Freie Universitat Berlin.

R. Srihari, C. Niu, and W. Li. 2001. A hybrid approach for named entity and sub-type tagging. In *ANLP*.

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

D. Wu, W. S. Lee, N. Ye, and H. L. Chieu. 2009. Domain adaptive bootstrapping for named entity recognition. In *EMNLP*.

J. Zhu, V. Uren, and E. Motta. 2005. Espotter: Adaptive named entity recognition for web browsing. In *WM*.

Collective Cross-Document Relation Extraction Without Labelled Data

Limin Yao Sebastian Riedel Andrew McCallum
University of Massachusetts, Amherst
{lmyao,riedel,mccallum}@cs.umass.edu

Abstract

We present a novel approach to relation extraction that integrates information across documents, performs global inference and requires no labelled text. In particular, we tackle relation extraction and entity identification jointly. We use distant supervision to train a factor graph model for relation extraction based on an existing knowledge base (Freebase, derived in parts from Wikipedia). For inference we run an efficient Gibbs sampler that leads to linear time joint inference. We evaluate our approach both for an in-domain (Wikipedia) and a more realistic out-of-domain (New York Times Corpus) setting. For the in-domain setting, our joint model leads to 4% higher precision than an isolated local approach, but has no advantage over a pipeline. For the out-of-domain data, we benefit strongly from joint modelling, and observe improvements in precision of 13% over the pipeline, and 15% over the isolated baseline.

1 Introduction

Relation Extraction is the task of predicting semantic relations over entities expressed in structured or semi-structured text. This includes, for example, the extraction of employer-employee relations mentioned in newswire, or protein-protein interactions expressed in biomedical papers. It also includes the prediction of entity types such as country, citytown or person, if we consider entity types as unary relations.

A particularly attractive approach to relation extraction is based on *distant supervision*.[1] Here in place of annotated text, only an existing knowledge base (KB) is needed to train a relation extractor (Mintz et al., 2009; Bunescu and Mooney, 2007; Riedel et al., 2010). The facts in the KB are heuristically aligned to an unlabelled training corpus, and the resulting alignment is the basis for learning the extractor.

Naturally, the predictions of a distantly supervised relation extractor will be less accurate than those of a supervised one. While facts of existing knowledge bases are inexpensive to come by, the heuristic alignment to text will often lead to noisy patterns in learning. When applied to unseen text, these patterns will produce noisy facts. Indeed, we find that extraction precision still leaves much room for improvement. This room is not as large as in previous work (Mintz et al., 2009) where target text and training KB are closely related. However, when we use the knowledge base Freebase (Bollacker et al., 2008) and the New York Times corpus (Sandhaus, 2008), we observe very low precision. For example, the precision of the top-ranked 50 `nationality` relation instances is only 28%.

On inspection, it turns out that many of the errors can be easily identified: they amount to violations of basic compatibility constraints between facts. In particular, we observe unsatisfied *selectional preferences* of relations towards particular entity types as types of their arguments. An example is the fact that the first argument of `nationality` is always a `person` while the second is a `country`. A simple way to address this is a pipeline: first predict entity types, and then condition on these when predicting relations. However, this neglects the fact that relations could as well be used to help entity type prediction.

[1] Also called *self training*, or *weak supervision*.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1013–1023,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

While there is some existing work on enforcing such constraints in a joint fashion (Roth and Yih, 2007; Kate and Mooney, 2010; Riedel et al., 2009), they are not directly applicable here. The difference is the amount of facts they take into account at the same time. They focus on single sentence extractions, and only consider very few interacting facts. This allows them to work with exact optimization techniques such as (Integer) Linear Programs and still remain efficient.[2] However, when working on a sentence level they fail to exploit the redundancy present in a corpus. Moreover, the fewer facts they consider at the same time, the lower the chance that some of these will be incompatible, and that modelling compatibility will make a difference.

In this work we present a novel approach that performs relation extraction across documents, enforces selectional preferences, and needs no labelled data. It is based on an undirected graphical model in which variables correspond to facts, and factors between them measure compatibility. In order to scale up, we run an efficient Gibbs-Sampler at inference time, and train our model using SampleRank (Wick et al., 2009). In practice this leads to a runtime behaviour that is linear in the size of the corpus. For example, 200,000 documents take less than three hours for training and testing.

For evaluation we consider two scenarios. First we follow Mintz et al. (2009), use Freebase as source of distant supervision, and employ Wikipedia as source of unlabelled text—we will call this an in-domain setting. This scenario is somewhat artificial in that Freebase itself is partially derived from Wikipedia, and in practice we cannot expect text and training knowledge base to be so close. Hence we also evaluate our approach on the New York Times corpus (out-of-domain setting).

For in-domain data we make the following finding. When we compare to an isolated baseline that makes no use of entity types, our joint model improves average precision by 4%. However, it does not outperform a pipelined system. In the out-of-domain setting, our collective model substantially outperforms both other approaches. Compared to the isolated baseline, we achieve a 15% increase in precision. With respect to the pipeline approach, the increase is 13%.

In the following we will first give some background information on relation extraction with distant supervision. Then we will present our graphical model as well as the inference and learning techniques we apply. After discussing related work, we present our empirical results and conclude.

2 Background

In this section we will introduce the terminology and concepts we use throughout the paper. We will also give a brief introduction to relation extraction, in particular in the context of distant supervision.

2.1 Relations

We seek to extract facts about entities. Example entities would be the company founder BILL GATES, the company MICROSOFT, and the country USA. A *relation* R is a set of tuples $\mathbf{c}$ over entities. We will follow (Mintz et al., 2009) and call the term $R(c_1, \ldots c_n)$ with $\mathbf{c} \in R$ a *relation instance*.[3] It denotes the membership of the tuple $\mathbf{c}$ in the relation R. For example, `founded` (BILL GATES, MICROSOFT) is a relation instance denoting that BILL GATES and MICROSOFT are related in the `founded` relation.

In the following we will always consider some set of *candidate tuples* C that may or may not be related. We define $C_n \subset C$ to be set of all n-ary tuples in C. Note that while our definition considers general n-nary relations, in practice we will restrict us to unary and binary relations C_1 and C_2.

Following previous work (Mintz et al., 2009; Zelenko et al., 2003; Culotta and Sorensen, 2004) we make one more simplifying assumption: every candidate tuple can be member of at most one relation.

2.2 Entity Types

An entity can be of one or several *entity types*. For example, BILL GATES is a `person`, and a company `founder`. Entity types correspond to the special case of relations with arity one, and will be treated as such in the following.

[2] The pyramid algorithm of Kate and Mooney (2010) may scale well, but it is not clear how to apply their scheme to cross-document extraction.

[3] Other commonly used terms are relational facts, ground facts, ground atoms, and assertions.

We care about entity types for two reasons. First, they can be important for downstream applications: if consumers of our extracted facts know the type of entities, they can find them more easily, visualize them more adequately, and perform operations specific to these types (write emails to persons, book a hotel in a city, etc.). Second, they are useful for extracting binary relations due to selectional preferences—see section 2.6.

2.3 Mentions

In natural language text spans of tokens are used to refer to entities. We call such spans *entity mentions*. Consider, for example, the following sentence snippet:

(1) Political opponents of President **Evo Morales** of **Bolivia** have in recent days stepped up...

Here "Evo Morales" is an entity mention of president EVO MORALES, and "Bolivia" a mention of the country BOLIVIA he is the president of.

People often express relations between entities in natural language texts by mentioning the participating entities in specific syntactic and lexical patterns. We will refer to any tuple of mentions of entities $(e_1, \ldots e_n)$ in a sentence as *candidate mention tuple*. If such a candidate expresses the relation R, then it is a *relation mention* of the relation instance $R(e_1, \ldots, e_n)$.

Consider again example 1. Here the pair of entity mentions ("Evo Morales", "Bolivia") is a *candidate mention tuple*. In fact, in this case the candidate is indeed a *relation mention* of the relation instance `nationality` (EVO MORALES, BOLIVIA).

2.4 Relation Extraction

We define the task of relation extraction as follows. We are given a corpus of documents and a set of target relations. Then we are asked to predict all relation instances I so that for each $R(\mathbf{c}) \in I$ there exists at least one relation mention in the given corpus.

The above definition covers a range of existing approaches by varying over what we define as target corpus. On one end, we have extractors that process text on a per sentence basis (Zelenko et al., 2003; Culotta and Sorensen, 2004). On the other end, we have methods that take relation mentions from several documents and use these as input features (Mintz et al., 2009; Bunescu and Mooney, 2007).

There is a compelling reason for performing relation extraction within a larger scope that considers mentions across documents: redundancy. Often facts are mentioned in several sentences and documents. Some of these mentions may be difficult to parse, or they use unseen patterns. But the more mentions we consider, the higher the probability that one does parse, and fits a pattern we have seen in the training data.

Note that for relation extraction that considers more than a single mention we have to solve the coreference problem in order to determine which mentions refer to the same entity. In the following we will assume that coreference clusters are provided by a preprocessing step.

2.5 Distant Supervision

In relation extraction we often encounter a lack of explicitly annotated text, but an abundance of structured data sources such as company databases or collaborative knowledge bases like Freebase. In order to exploit this, many approaches use simple but effective heuristics to align existing facts with unlabelled text. This labelled text can then be used as training material of a supervised learner.

One heuristic is to assume that each candidate mention tuple of a training fact is indeed expressing the corresponding relation (Bunescu and Mooney, 2007). Mintz et al. (2009) refer to this as the *distant supervision assumption*.

Clearly, this heuristic can fail. Let us again consider the `nationality` relation between EVO MORALES and BOLIVIA. In an 2007 article of the New York Times we find this relation mention candidate:

(2) ...the troubles faced by **Evo Morales** in **Bolivia**...

This sentence does not directly express that EVO MORALES is a citizen of BOLIVIA, and hence violates the distant supervision assumption. The problem with this observation is that at training time we may learn a relatively large weight for the feature "<Entity1> in <Entity2>" associated with nationality. When testing our model we then encounter a sentence such as

(3) Arrest Warrant Issued for **Richard Gere** in **India**.

that leads us to extract that RICHARD GERE is a citizen of INDIA.

2.6 Global Consistency of Facts

As discussed above, distant supervision can lead to noisy extractions. However, such noise can often be easily identified by testing how compatible the extracted facts are to each other. In this work we are concerned with a particular type of compatibility: selectional preferences.

Relations require, or prefer, their arguments to be of certain types. For example, the nationality relation requires the first argument to be a person, and the second to be a country. On inspection, we find that these preferences are often not satisfied in a baseline distant supervision system akin to Mintz et al. (2009). This often results from patterns such as "<Entity1> in <Entity2>" that fire in many cases where <Entity2> is a location, but not a country.

3 Model

Our observations in the previous section suggest that we should (a) explicitly model compatibility between extracted facts, and (b) integrate evidence from several documents to exploit redundancy. In this work we choose a Conditional Random Field (CRF) to achieve this. CRFs are a natural fit for this task: They allow us to capture correlations in an explicit fashion, and to incorporate overlapping input features from multiple documents.

The hidden output variables of our model are $\mathbf{Y} = (Y_\mathbf{c})_{\mathbf{c} \in C}$. That is, we have one variable $Y_\mathbf{c}$ for each candidate tuple $\mathbf{c} \in C$. This variable can take as value any relation in C with the same arity as $\mathbf{c}$. See example relation variables in figure 1.

The observed input variables $\mathbf{X}$ consists of a family of variables $\mathbf{X_c} = \left(\mathbf{X}_\mathbf{c}^1, \dots \mathbf{X}_\mathbf{c}^m\right)_{m \in M}$ for each candidate tuple $\mathbf{c}$. Here $\mathbf{X}_\mathbf{c}^i$ stores relevant observations we make for the i-th candidate mention tuple of $\mathbf{c}$ in the corpus. For example, $\mathbf{X}^1_{\text{BILL GATES,MICROSOFT}}$ in figure 1 would contain, among others, the pattern "[M2] was founded by [M1]".

3.1 Factor Templates

Our conditional probability distribution over variables $\mathbf{X}$ and $\mathbf{Y}$ is defined using using a set $\mathcal{T}$ of *factor templates*. Each template $T_j \in \mathcal{T}$ defines a set of factors $\{(\mathbf{y}_i, \mathbf{x}_i)\}$, a set K_j of feature indices, parameters $\left\{\theta_k^j\right\}_{k \in K_j}$ and feature functions $\left\{f_k^j\right\}_{k \in K_j}$. Together they define the following conditional distribution:

$$p\left(\mathbf{y}|\mathbf{x}\right) = \frac{1}{Z_\mathbf{x}} \prod_{T_j \in \mathcal{T}} \prod_{(\mathbf{y}_i, \mathbf{x}_i) \in T_j} e^{\sum_{k \in K_j} \theta_k^j f_k^j(\mathbf{y}_i, \mathbf{x}_i)} \tag{4}$$

In our case the set $\mathcal{T}$ consists of four templates we will describe below. We construct this graphical model using FACTORIE (McCallum et al., 2009), a probabilistic programming language that simplifies the construction process, as well as inference and learning.

3.1.1 Bias Template

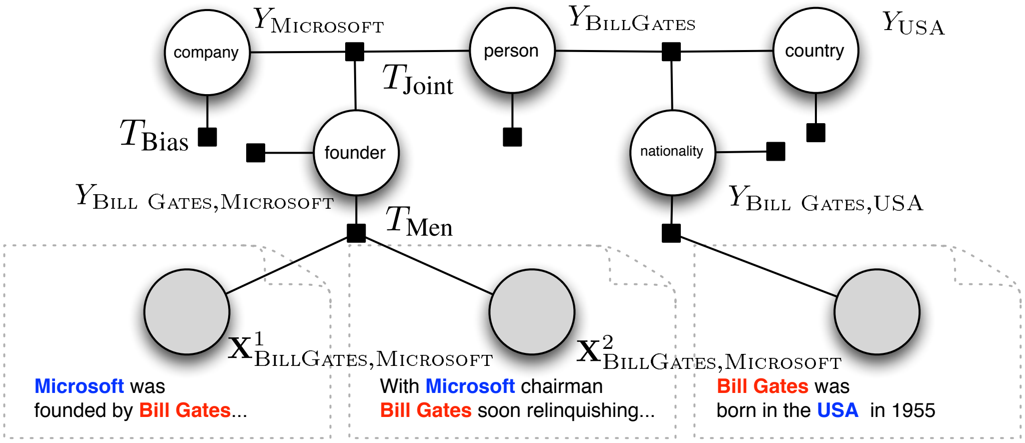
We use a bias template T_{Bias} that prefers certain relations *a priori* over others. When the template is unrolled, it creates one factor per variable $Y_\mathbf{c}$ for candidate tuple $\mathbf{c} \in C$. The template also consists of one weight θ_r^{Bias} and feature function f_r^{Bias} for each possible relation r. f_r^{Bias} fires if the relation associated with tuple $\mathbf{c}$ is r.

3.1.2 Mention Template

In order to extract relations from text, we need to model the correlation between relation instances and their mentions in text. For this purpose we define the template T_{Men} that connects each relation instance variable $Y_\mathbf{c}$ with its observed mention variables $\mathbf{X_c}$. Crucially, this template gathers mentions from multiple documents, and enables us to exploit redundancy.

The feature functions of this template are taken from Mintz et al. (2009). This includes features that inspect the lexical content between entity mentions in the same sentence, and the syntactic path between them. One example is

$$f_{101}^{\text{Men}}\left(y_\mathbf{c}, \mathbf{x_c}\right) \overset{\text{def}}{=} \begin{cases} 1 & y_\mathbf{c} = \texttt{founded} \wedge \exists i \text{ with} \\ & \text{"M2 was founded by M1"} \in \mathbf{x}_\mathbf{c}^i \\ 0 & \text{otherwise} \end{cases}.$$



company
$Y_{\text{MICROSOFT}}$
person
$Y_{\text{BILLGATES}}$
country
Y_{USA}
T_{Joint}
T_{Bias}
founder
nationality
$Y_{\text{BILL GATES,MICROSOFT}}$
T_{Men}
$Y_{\text{BILL GATES,USA}}$
$\mathbf{X}^1_{\text{BILLGATES,MICROSOFT}}$
$\mathbf{X}^2_{\text{BILLGATES,MICROSOFT}}$
Microsoft was founded by Bill Gates...
With Microsoft chairman Bill Gates soon relinquishing...
Bill Gates was born in the USA in 1955

Figure 1: Factor Graph of our model that captures selectional preferences and functionality constraints. For readability we only label a subsets of equivalent variables and factors. Note that the graph shows an example assignment to variables.

It tests whether for any mentions of the candidate tuple the phrase "founded by" appears between the mentions of the argument entities.

3.1.3 Selectional Preferences Templates

To capture the correlations between entity types and relations the entities participate in, we introduce the template T_{Joint}. It connects a relation instance variable $Y_{e_1,\ldots,e_n}$ to the individual entity type variables $Y_{e_1},\ldots,Y_{e_n}$. To measure the compatibility between relation and entity variables, we use one feature $f^{\text{Joint}}_{r,t_1\ldots t_a}$ (and weight $\theta^{\text{Joint}}_{r,t_1\ldots t_a}$) for each combination of relation and entity types $r, t_1 \ldots t_a$.

$f^{\text{Joint}}_{r,t_1\ldots t_a}$ fires when the factor variables are in the state $r, t_1 \ldots t_a$. For example, $f^{\text{Joint}}_{\texttt{founded},\texttt{person},\texttt{company}}$ fires if Y_{e_1} is in state `person`, Y_{e_2} in state `company`, and Y_{e_1,e_2} in state `founded`.

We also add a template T_{Pair} that measures the pairwise compatibility between the relation variable $Y_{e_1,\ldots,e_a}$ and each entity variable Y_{e_i} in isolation. Here we use features $f^{\text{Pair}}_{i,r,t}$ that fire if e_i is the i-th argument of $\mathbf{c}$, has the entity type t and the candidate tuple $\mathbf{c}$ is labelled as instance of relation r. For example, $f^{\text{Pair}}_{1,\texttt{founded},\texttt{person}}$ fires if Y_{e_1}(argument $i = 1$) is in state `person`, and Y_{e_1,e_2} in state `founded`, regardless of the state of Y_{e_2}.

3.2 Inference

There are two types of inference we have to perform: sampling from the posterior during training (see section 3.3), and finding the most likely configuration (aka MAP inference). In both settings we employ a Gibbs sampler (Geman and Geman, 1990) that randomly picks a variable $Y_{\mathbf{c}}$ and samples its relation value conditioned on its Markov Blanket. At test time we decrease the temperature of our sampler in order to find an approximation of the MAP solution.

3.3 Training

Most learning methods need to calculate the model expectations (Lafferty et al., 2001) or the MAP configuration (Collins, 2002) before making an update to the parameters. This step of inference is usually the bottleneck for learning, even when performed approximately.

SampleRank (Wick et al., 2009) is a rank-based learning framework that alleviates this problem by performing parameter updates *within* MCMC inference. Every pair of consecutive samples in the MCMC chain is ranked according to the model and the ground truth, and the parameters are updated when the rankings disagree. This update can follow different schemes, here we use MIRA (Crammer and Singer, 2003). This allows the learner to acquire more supervision per instance, and has led to efficient training for models in which inference is expensive and generally intractable (Singh et al., 2009).

4 Related Work

Distant Supervision Learning to extract relations by using distant supervision has raised much interest in recent years. Our work is inspired by Mintz et al. (2009) who also use Freebase as distant supervision source. We also heuristically align our knowledge base to text by making the distant supervision assumption (Bunescu and Mooney, 2007; Mintz et al., 2009). However, in contrast to these previous approaches, and other related distant supervision methods (Craven and Kumlien, 1999; Weld et al., 2009; Hoffmann et al., 2010), we perform relation extraction collectively with entity type prediction.

Schoenmackers et al. (2008) use entailment rules on assertion extracted by TextRunner to increase recall. They also perform cross-document probabilistic inference based on Markov Networks. However, they do not infer the types of entities and work in an open IE setting.

Selectional Preferences In the context of supervised relation extraction, selectional preferences have been applied. For example, Roth and Yih (2007) have used Linear Programming to enforce consistency between entity types and extracted relations. Kate and Mooney (2010) use a pyramid parsing scheme to achieve the same. Riedel et al. (2009) use Markov Logic to model interactions between event-argument relations for biomedical event extraction. However, their work is (a) supervised, and (b) performs extraction on a per-sentence basis.

Carlson et al. (2010) also use selectional preferences. However, instead of exploiting them for training a graphical model using distant supervision, they use selectional preferences to improve a bootstrapping process. Here in each iteration of bootstrapping, extracted facts that violate compatibility constraints will not be used to generate additional patterns in the next iteration.

5 Experiments

We set up experiments to answer the following questions: (i) Does the explicit modelling of selectional preferences improve accuracy? (ii) Can we also perform joint entity and relation extraction in a pipeline and achieve similar results? (iii) How does our cross-document approach scale?

To answer these questions we carry out experiments on two data sets, Wikipedia and New York Times articles, and use Freebase as distant supervision source for both.

5.1 Experimental Setup

We follow Mintz et al. (2009) and perform two types of evaluation: held-out and manual. In both cases we have a training and a test corpus of documents, and training and test sets of entities. For held-out evaluation we split the set of entities in Freebase into training and test sets. For manual evaluation we use all Freebase entities during training. For testing we use all entities that appear in the test document corpus.

For both training and testing we then choose the candidate tuples C that may or may not be relation instances. To pick the entities C_1 we want to predict entity types for, we choose all entities that are mentioned at least once in the train/test corpus. To pick the entity pairs C_2 that we want to predict the relations of, we choose those that appear at least once together in a sentence.

The set of candidates C will contain many tuples which are not related in any Freebase relations. For efficiency, we filter out a large fraction of these *negative* candidates for training. The number of negative examples we keep is chosen to be about 10 times the number of positive candidates. This number stems from trading-off the accuracy it leads to and the increased training time it requires.

For both manual and held-out evaluation we rank extracted test relation instances in the MAP state of the network. This state is found by sampling 20 iterations with a low temperature of 0.00001. The ranking is done according to the log linear score that the assigned relation for a candidate tuple gets from the factors in its Markov Blanket. For optimal performance, the score is normalized by the number of relation mentions.

For manual evaluation we pick the top ranked 50 relation instances for the most frequent relations. We ask three annotators to inspect the mentions of these relation instances to decide whether they are correct. Upon disagreement, we use majority vote. To summarize precisions across relations, we take their average, and their average weighted by the proportion of predicted instances for the given relation.

5.1.1 Data preprocessing

We preprocess our textual data as follows: We first use the Stanford named entity recognizer (Finkel et al., 2005) to find entity mentions in the corpus. The NER tagger segments each document into sentences and classifies each token into four categories: PERSON, ORGANIZATION, LOCATION and NONE. We treat consecutive tokens which share the same category as single entity mention. Then we associate these mentions with Freebase entities. This is achieved by performing a string match between entity mention phrases and the canonical names of entities as present in Freebase.

For each candidate tuple **c** with arity 2 and each of its mention tuples i we extract a set of features $\mathbf{X}_{\mathbf{c}}^{i}$ similar to those used in (Mintz et al., 2009): lexical, Part-Of-Speech (POS), named entity and syntactic features, i.e. features obtained from the dependency parsing tree of a sentence. We use the openNLP POS tagger[4] to obtain POS tags and employ the MaltParser (Nivre et al., 2004) for dependency parsing. For candidate tuples with arity 1 (entity types) we use the following features: the entity's word form, the POS sequence, the head of the entity in the dependency parse tree, the Stanford named entity tag, and the left and right words to the current entity mention phrase.

5.1.2 Configurations

We apply the following configurations of our factor graphs. As our baseline, and roughly equivalent to previous work (Mintz et al., 2009), we pick the templates T_{Bias} and T_{Men}. These describe a fully disconnected graph, and we will refer to this configuration as *isolated*. Next, we add the templates T_{Joint} and T_{Pair} to model selectional preferences, and refer to this setting as *joint*.

In addition, we evaluate how well selectional preferences can be captured with a simple *pipeline*. For this pipeline we first train an isolated system for entity type prediction. Then we use the output of the entity type prediction system as input for the relation extraction system.

5.1.3 Entity types and Relation types

Freebase contains many relation types and only a subset of those relation types occur frequently in the corpus. Since classes with very few training instances are generally hard to learn, we restrict ourselves to the 54 most frequently mentioned relations. These include, for example, `nationality`, `contains`, `founded` and `place_of_birth`. Note that we convert two Freebase non-binary temporal relations to binary relations: `employment_tenure` and `place_lived`. In both cases we simply disregard the temporal information in the Freebase data.

As our main focus is relation extraction, we restrict ourselves to entity types compatible with our selected relations. To this end we inspect the Freebase schema information provided for each relation, and include those entity types that are declared as arguments of our relations. This leads to 10 entity types including `person`, `citytown`, `country`, and `company`.

Note that a Freebase entity can have several types. We pick one of these by choosing the most specific one that is a member of our entity type subset, or `MISC` if no such member exists.

5.2 Wikipedia

In our first set of experiments we train and test using Wikipedia as the text corpus. This is a comparatively easy scenario because the facts in Freebase are partly derived from Wikipedia, hence there is an increased chance of properly aligning training facts and text. This is similar to the setting of Mintz et al. (2009).

5.2.1 Held Out Evaluation

We split 1,300,000 Wikipedia articles into training and test sets. Table 1 shows the statistics for this split. The last row provides the number of negative relation instances (candidates which are not related according to Freebase) associated with each data set.

Figure 2 shows the precision-recall curves of relation extraction for held-out data of various configurations. We notice a slight advantage of the joint approach in the low recall area. Moreover, the joint model predicts more relation instances, as can be seen by its longer line in the graph.

For higher recall, the joint model performs slightly worse. On closer inspection, we find that

[4] available at http://opennlp.sourceforge.net/

	Wikipedia		NYT	
	Train	Test	Train	Test
#Documents	900K	400K	177K	39K
#Entities	213K	137K	56K	27K
#Positive	36K	24K	5K	2K
#Negative	219K	590K	64K	94K

Table 1: The statistics of held-out evaluation on Wikipedia and New York Times.

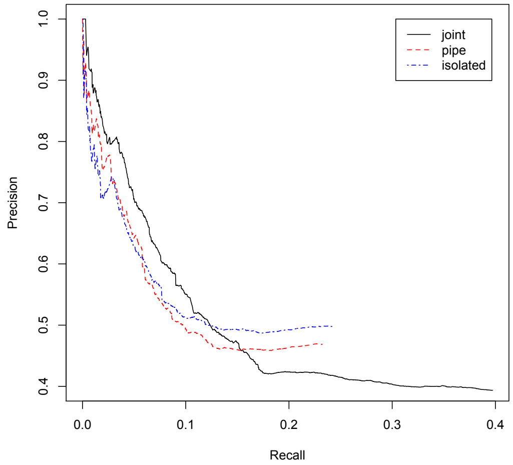


Figure 2: Precision-recall curves for various setups in Wikipedia held-out setting.

this observation is somewhat misleading. Many of the predictions of the joint model are not in the held-out test set derived from Freebase, but nevertheless correct. Hence, to understand if one system really outperforms another, we need to rely on manual evaluation.

Note that the figure only considers binary relations—for entity types all configurations perform similarly.

5.2.2 Manual Evaluation

As mentioned above, held-out evaluation in this context suffers from false negatives in Freebase. Table 2 therefore shows the results of our manual evaluation. They are based on the average, and weighted average, of the precisions for the relation instances of the most frequent relations. We notice that here all systems perform comparably for weighted average precision. For average precision we see an advantage for both the pipeline and the joint model over the isolated system.

	Isolated	Pipeline	Joint
Wikipedia	0.82	0.87	0.86
Wikipedia (w)	0.95	0.94	0.95
NYT	0.63	0.65	0.78
NYT (w)	0.78	0.82	0.94

Table 2: Average and weighted (w) average precision over frequent relations for New York Times and Wikipedia data, based on manual evaluation.

One reason for similar weighted average precisions is the fact that all approaches accurately predict a large number of `contains` instances. This is due to very regular and simple patterns in Wikipedia. For example, most articles on towns start with “A is a municipality in the district of B in C, D.” For these sentences, the relative position of two location mentions is a very good predictor of `contains`. When used as a feature, it leads to high precision for all models. And since `contains` instances are most frequent, and we take the weighted average, results are generally close to each other.

To summarize: in this in-domain setting, modelling compatibility between entity types and relations helps to improve average precision, but not weighted average precision. This holds for both the joint and the pipeline model. However, we will see how this changes substantially when moving to an out-of-domain scenario.

5.3 New York Times

For our second set of experiments we use New York Times data as training and test corpora. As we argued before, this is expected to be the more difficult—and more realistic—scenario.

5.3.1 Held-out Evaluation

We choose all articles of the New York times during 2005 and 2006 as training corpus. As test corpus we use the first 6 months of 2007.

Figure 3 shows precision-recall curves for our various setups. We see that jointly modelling entity

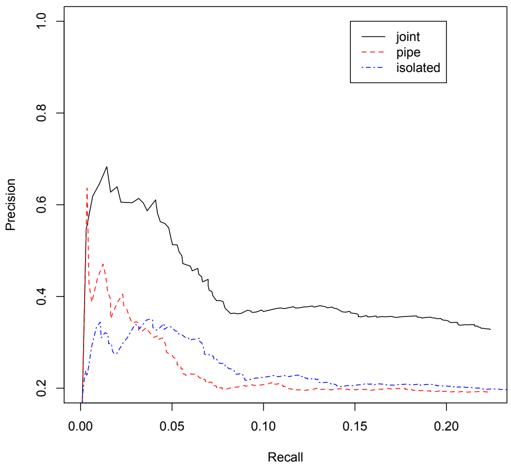


Figure 3: Precision-recall curves for various setups in New York Times held-out setting.

types and relations helps to improve precision.

Due to the smaller overlap between Freebase and NYT data, figure 3 also has to be taken with more caution. The systems may predict correct relation instances that just do not appear in Freebase. Hence manual evaluation is even more important.

When evaluating entity precision we find that for both models it is about 84%. This raises the question why the joint entity type and relation extraction model outperforms the pipeline on relations. We take a close look at the entities which participate in relations and find that joint model performs better on most entity types, for example, country and citytown. We also look at the relation instances which are predicted by both systems and find that the joint model does predict correct entity types when the pipeline mis-predicts. And exactly these mis-predictions lead the pipeline astray. Considering binary relation instances where the pipeline fails but the joint model does not, we observe an entity precision of 76% for the pipeline and 86% for our joint approach. The joint model fails to correctly predict some entity types that the pipeline gets right, but these tend to appear in contexts where relation instances are easy to extract without considering entity types.[5]

Relation Type	Iso.	Pipe	Joint
contains	0.92	0.98	0.96
nationality	0.28	0.64	0.82
plc_lived	0.88	0.70	0.96
plc_of_birth	0.32	0.20	0.25
works_for	0.96	0.98	0.98
plc_of_death	0.24	0.40	0.42
children	1.00	0.92	0.98
founded	0.42	0.34	0.71

Table 3: Precision at 50 for the most frequent relations on New York Times

5.3.2 Manual Evaluation

Manually evaluated precision for New York Times data can be seen in table 2. In contrast to the Wiki setting, here modelling entity types and relations jointly makes a substantial difference. For average precision, our joint model improves over the isolated baseline by 15%, and over the pipeline by 13%. Similar improvements can be observed for weighted average precision.

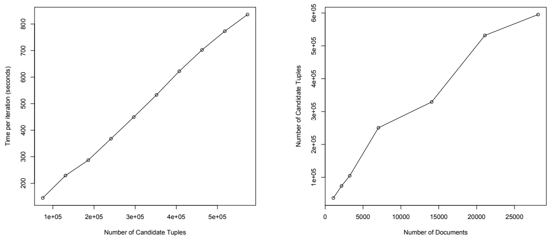
Let us look at a break-down of precisions with respect to different relations shown in table 3. We see dramatic improvements for nationality and founded when applying the joint model. Note that the nationality relation takes a larger part in the predicted relation instances of the joint model and hence contributes significantly to the weighted average precision.

5.4 Scalability

We propose to perform joint inference for large scale information extraction. An obvious concern in this scenario is scalability. In practice we find that inference (and hence learning) in our model scales linearly with the number of candidate tuples. This can be seen in figure 4a. It is to be expected since the number of candidates equals the number of variables the sampler has to process in each iteration.

The above observation also means that our approach scales linearly with corpus size. To illustrate

[5] Note that our learned preferences are soft, and hence can be violated in case of wrong entity type predictions.



(a) CPU time

(b) Candidate tuples

Figure 4: CPU time for one iteration per candidate tuple, and candidate tuples per document.

this, figure 4b shows how the number of candidates scales with the number of documents. Again we observe a linear behavior. Since both are linear, we can say that our joint approach is linear in the number of documents.

Total training and test times are moderate, too. For example, the held-out experiments with 200,000 NYT documents finish within three hours.

6 Conclusion

This paper presents a novel approach to extracting relational facts from text. Akin to previous work in relation extraction with distant supervision, we require no annotated text. However, instead extracting facts in isolation, we model interactions between facts in order to improve precision. In particular, we capture selectional preferences of relations. These preferences are modelled in a cross-document fashion using a large scale factor graph. We show inference and learning can be efficiently performed in linear time by Gibbs Sampling and SampleRank. When applied to out-of-domain text, this approach leads to a 15% increase in precision over an isolated baseline, and a 13% improvement over a pipelined system.

A crucial aspect of our approach is its extensibility. Since it is exclusively framed in terms of an undirected graphical model, it is conceptually easy to extend it to other types of compatibilities, such as functionality constraints. It could also be extended to tackle coreference resolution. Eventually we seek to model the complete process of the automatic construction of KB within this framework, and capture dependencies between extractions in a joint and principled fashion. As we have seen here, in particular when learning is less supervised and extractions are noisy, capturing such interactions is paramount.

Acknowledgements

This work was supported in part by the Center for Intelligent Information Retrieval, in part by The Central Intelligence Agency, the National Security Agency and National Science Foundation under NSF grant #IIS-0326249, and in part by UPenn NSF medium IIS-0803847. The University of Massachusetts also gratefully acknowledges the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, New York, NY, USA. ACM.

Razvan C. Bunescu and Raymond J. Mooney. 2007. Learning to extract relations from the web using minimal supervision. In *Proceedings of the 45rd Annual Meeting of the Association for Computational Linguistics (ACL '07)*.

Andrew Carlson, Justin Betteridge, Richard Wang, Estevam Hruschka, and Tom Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Third ACM International Conference on Web Search and Data Mining (WSDM '10)*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical methods in natural language processing (EMNLP '02)*, volume 10, pages 1–8.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

M. Craven and J. Kumlien. 1999. Constructing biological knowledge-bases by extracting information from text sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 77–86, Germany.

Aron Culotta and Jeffery Sorensen. 2004. Dependency tree kernels for relation extraction. In *42nd Annual Meeting of the Association for Computational Linguistics*, Barcelona, Spain.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL '05)*, pages 363–370, June.

S. Geman and D. Geman. 1990. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. pages 452–472.

Raphael Hoffmann, Congle Zhang, and Daniel S. Weld. 2010. Learning 5000 relational extractors. In *ACL*.

Rohit J. Kate and Raymond J. Mooney. 2010. Joint entity and relation extraction using card-pyramid parsing. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL' 10)*.

John D. Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning (ICML)*.

Andrew McCallum, Karl Schultz, and Sameer Singh. 2009. Factorie: Probabilistic programming via imperatively defined factor graphs. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1249–1257.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL '09)*, pages 1003–1011. Association for Computational Linguistics.

J. Nivre, J. Hall, and J. Nilsson. 2004. Memory-based dependency parsing. In *Proceedings of CoNLL*, pages 49–56.

Sebastian Riedel, Hong-Woo Chun, Toshihisa Takagi, and Jun'ichi Tsujii. 2009. A markov logic approach to bio-molecular event extraction. In *Proceedings of the Natural Language Processing in Biomedicine NAACL 2009 Workshop (BioNLP '09)*, pages 41–49.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD '10)*.

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.

Evan Sandhaus, 2008. *The New York Times Annotated Corpus*. Linguistic Data Consortium, Philadelphia.

Stefan Schoenmackers, Oren Etzioni, and Daniel S. Weld. 2008. Scaling textual inference to the web. In *EMNLP '08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 79–88, Morristown, NJ, USA. Association for Computational Linguistics.

Sameer Singh, Karl Schultz, and Andrew McCallum. 2009. Bi-directional joint inference for entity resolution and segmentation using imperatively-defined factor graphs. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, pages 414–429.

Daniel S. Weld, Raphael Hoffmann, and Fei Wu. 2009. Using wikipedia to bootstrap open information extraction. In *ACM SIGMOD Record*.

Michael Wick, Khashayar Rohanimanesh, Aron Culotta, and Andrew McCallum. 2009. Samplerank: Learning preferences from atomic gradients. In *Neural Information Processing Systems (NIPS), Workshop on Advances in Ranking*.

Dimitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *JMLR*, 3(6):1083 – 1106.

Automatic Detection and Classification of Social Events

Apoorv Agarwal
Department of Computer Science
Columbia University
New York, U.S.A.
apoorv@cs.columbia.edu

Owen Rambow
CCLS
Columbia University
New York, U.S.A.
rambow@ccls.columbia.edu

Abstract

In this paper we introduce the new task of *social event* extraction from text. We distinguish two broad types of social events depending on whether only one or both parties are aware of the social contact. We annotate part of Automatic Content Extraction (ACE) data, and perform experiments using Support Vector Machines with Kernel methods. We use a combination of structures derived from phrase structure trees and dependency trees. A characteristic of our events (which distinguishes them from ACE events) is that the participating entities can be spread far across the parse trees. We use syntactic and semantic insights to devise a new structure derived from dependency trees and show that this plays a role in achieving the best performing system for both social event detection and classification tasks. We also use three data sampling approaches to solve the problem of data skewness. Sampling methods improve the F1-measure for the task of relation detection by over 20% absolute over the baseline.

1 Introduction

This paper introduces a novel natural language processing (NLP) task, *social event extraction*. We are interested in this task because it contributes to our overall research goal, which is to extract a social network from written text. The extracted social network can be used for various applications such as summarization, question-answering, or the detection of main characters in a story. For example, we manually extracted the social network of characters in *Alice in Wonderland* and ran standard social network analysis algorithms on the network. The most influential characters in the story were correctly detected. Moreover, characters occurring in a scene together were given same social roles and positions. Social network extraction has recently been applied to literary theory (Elson et al., 2010) and has the potential to help organize novels that are becoming machine readable.

We take a "social network" to be a network consisting of individual human beings and groups of human beings who are connected to each other by the virtue of participating in social events. We define social events to be events that occur between people where at least one person is aware of the other and of the event taking place. For example, in the sentence *John talks to Mary*, entities John and Mary are aware of each other and the talking event. In the sentence *John thinks Mary is great*, only John is aware of Mary and the event is the thinking event. In the sentence *Rabbit ran by Alice* there is no evidence about the cognitive states of Rabbit and Alice (because the Rabbit could have run by Alice without any one of them noticing each other). A text can describe a social network in two ways: explicitly, by stating the type of relationship between two individuals (e.g. husband-wife), or implicitly, by describing an event which creates or perpetuates a social relationship (e.g. *John talked to Mary*). We will call these types of events *social events*. We define two types of social events: **interaction**, in which both parties are aware of the social event (e.g., a conversation), and **observation**, in which only one party is aware of the interaction (e.g., thinking about or

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1024–1034,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

spying on someone). Note that the notion of cognitive state is crucial to our definition. This paper is the first attempt to detect and classify social events present in text.

Our task is different from related tasks, notably from the Automated Content Extraction (ACE) relation and event extraction tasks because the events are different (they are a class of events defined through the effect on participants' cognitive state), and the linguistic realization is different. Mentions of entities[1] engaged in a social event are often quite distant from each other in the sentence (unlike in ACE relations where about 70% of relations are local, in our social event annotation, only 25% of the events are local. In fact, the average number of words between entities participating in any social event is 9.)

We use tree kernel methods (on structures derived from phrase structure trees and dependency trees) in conjunction with Support Vector Machines (SVMs) to solve our tasks. For the design of structures and type of kernel, we take motivation from a system proposed by Nguyen et al. (2009) which is a state-of-the-art system for relation extraction. Data skewness turns out to be a big challenge for the task of relation detection since there are many more pairs of entities without a relation as compared to pairs of entities that have a relation. In this paper we discuss three data sampling techniques that deal with this skewness and allow us to gain over 20% in F1-measure over our baseline system. Moreover, we introduce a new sequence kernel that outperforms previously proposed sequence kernels for the task of social event detection and plays a role to achieve the best performing system for the task of social event detection and classification.

The paper is structured as follows. In Section 2, we compare our work to existing work, notably the ACE extraction literature. In Section 3, we present our task in detail, and explain how we annotated our corpus. We also show why this is a novel task, and how it is different from the ACE extraction tasks. We then discuss kernel methods and the structures we use, and introduce our new structure in Section 4. In Section 5, we present the sampling methods used for experiments. In Section 6 we present our experiments and results for social event detection and social event classification tasks. We conclude in Section 7 and mention our future direction of research.

2 Literature Survey

There has not been much work in developing techniques for ACE event extraction as compared to ACE relation extraction. The most salient work for event extraction is Grishman et al. (2005) and Ji and Grishman (2008). To solve the task for event extraction, Grishman et al. (2005) mainly use a combination of pattern matching and statistical modeling techniques. They extract two kinds of patterns: 1) the sequence of constituent heads separating anchor and its arguments and 2) a predicate argument subgraph of the sentence connecting anchor to all the event arguments. In conjunction they use a set of Maximum Entropy based classifiers for 1) Trigger labeling, 2) Argument classification and 3) Event classification. Ji and Grishman (2008) further exploit a correlation between senses of verbs (that are triggers for events) and topics of documents.

Our work shares some similarities. However, instead of building different classifiers, we use kernel methods with SVMs that "naturally" combine various patterns. The structures we use for kernel methods are a super-set of the patterns used by Grishman et al. (2005). Moreover, in our work, we take gold annotation for entity mentions, and do not deal with the task of named entity detection or resolution. Finally, our social events are a broad class of event types, and they involve linguistic expressions for expressing interactions and cognition that do not seem to have a correlation with the topics of documents.

There has been much work in extracting ACE relations. The supervised approaches used for relation extraction can broadly be divided into three main categories: 1) feature-based approaches 2) kernel-based approaches and 3) a combination of feature and kernel based approaches. The state-of-the-art feature based approach is that of GuoDong et al. (2005). They use diverse lexical, syntactic and semantic knowledge for the task. The lexical features they use are words between, before, and after target entity mentions, the type of entity (Person, Organization etc.), the type of mention (named, nominal or pronominal) and a feature called overlap

[1] An *entity mention* is a reference of an entity in text. Also, we use *entities* and *people* interchangeably since the only entities we are interested in are people or groups of people.

that counts the number of other entity mentions and words between the target entities. To incorporate syntactic features they use features extracted from base phrase chunking, dependency trees and phrase structure trees. To incorporate semantic features, their approach uses resources like a country list and WordNet. GuoDong et al. (2005) report that 70% of the entities are embedded within each other or separated by just one word. This is a major difference to our task because most of our relations span over a long distance in a sentence.

Collins and Duffy (2002) are among the earliest researchers to propose the use of tree kernels for various NLP tasks. Since then kernels have been used for the task of relation extraction (Zelenko et al., 2002; Zhao and Grishman, 2005; Zhang et al., 2006; Moschitti, 2006b; Nguyen et al., 2009). For an excellent review of these techniques, see Nguyen et al. (2009). In addition, there has been some work that combines feature and kernel based methods (Harabagiu et al., 2005; Culotta and Jeffrey, 2004; Zhou et al., 2007). Apart from using kernels over dependency trees, Culotta and Jeffrey (2004) incorporate features like words, part of speech (POS) tags, syntactic chunk tag, entity type, entity level, relation argument and WordNet hypernym. Harabagiu et al. (2005) leverage this approach by adding more semantic feature derived from semantic parsers for FrameNet and PropBank. Zhou et al. (2007) use a context sensitive kernel in conjunction with features they used in their earlier publication (GuoDong et al., 2005). However, we take an approach similar to Nguyen et al. (2009). This is because it incorporates many of the features suggested in feature-based approaches by using combinations of various structures derived from phrase structure trees and dependency trees. In addition we use data sampling techniques to deal with the problem of data skewness. We not only try the structures suggested by Nguyen et al. (2009) but also introduce a new sequence structure on dependency trees. We discuss their structures and kernel method in detail in Section 4.

3 Social Event Annotation Data

3.1 Social Event Annotation

There has been much work in the past on annotating entities, relations and events in free text, most notably the ACE effort (Doddington et al., 2004). We leverage this work by annotating social events on the English part of ACE 2005 Multilingual Training Data[2] that has already been annotated for entities, relations and events. In Agarwal et al. (2010), we introduce a comprehensive set of social events which are conceptually different from the event annotation that already exists for ACE. Since our annotation task is complex and layered, in Agarwal et al. (2010) we present confusion matrices, Cohen's Kappa, and F-measure values for each of the decision points that the annotators go through in the process of selecting a type and subtype for an event. Our annotation scheme is reliable, achieving a moderate kappa for relation detection (0.68) and a high kappa for relation classification (0.86). We also achieve a high global agreement of 69.7% using a measure which is inspired by Automated Content Extraction (ACE) inter-annotator agreement measure. This compares favorably to the ACE annotation effort.

Following are the two broad types of social events that were annotated:

Interaction event (INR): When both entities participating in an event are aware of each other and of the social event, we say they have an INR relation. Consider the following Example (1).

(1) [Toujan Faisal], 54, {said} [she] was {informed} of the refusal by an [Interior Ministry committee] overseeing election preparations. INR

As is intuitive, if one person *informs* the other about something, both have to be cognizant of each other and of the *informing* event in which they are both participating.

Observation event (OBS): When only one person (out of the two people that are participating in an event) is aware of the other and of the social event, we say they have an OBS relation. Of the type OBS, there are three subtypes: Physical Proximity (PPR), Perception (PCR) and Cognition (COG). PPR requires that one entity can observe the other entity in real time not through a broadcast medium, in contrast to the subtype PCR, where one entity observes the other through media (TV, radio, magazines etc.) Any other observation event that is not PPR or PCR

[2] Version: 6.0, Catalog number: LDC2005E18

is COG. Consider the aforementioned Example (1). In this sentence, the event *said* marks a COG relation between **Toujan Faisal** and the **committee**. This is because, when one person talks about another person, the other person must be present in the first person's cognitive state without any requirement on physical proximity or external medium.

As the annotations revealed, PPR and PCR occurred only twice and once, respectively, in the part of ACE corpus we annotated. (They occur more frequently in another genre we are investigating such as literary texts.) We omit these extremely low-frequency categories from our current study; in this paper we build classifiers to detect and classify only INR and COG events.

3.2 Comparison Between Social Events and ACE Annotations

The ACE effort is about entity, relation and event annotation. We use their annotations for entity types PER.Individual and PER.Group and add our social event annotations. Our event annotations are different from ACE event annotations because we annotate text that expresses the cognitive states of the people involved, or allows the annotator to infer it. Therefore, at the top level of classification we differentiate between events in which only one entity is cognizant of the other (observation) versus events when both entities are cognizant of each other (interaction). This distinction is, we believe, novel in event or relation annotation.

Now we present statistics and examples to make clear how our annotations are different from ACE event annotations. The statistics are based on 62 documents from the ACE corpus. These files contain a total of 212 social events. We found a total of 63 candidate ACE events that had at least two Person entities involved. Out of these 63 candidate events, 54 match our annotations. The majority of social events that match the ACE events are of type INR. On analysis, we found that most of these correspond to the ACE event type CONTACT. Specifically, the "meeting" event, which is an ACE CONTACT event and an INR event according to our definition, is the major cause of overlap. However, our type INR has a broader definition than ACE type CONTACT. For example, in Example 1, we recorded an INR event between **Toujan Faisal** and **committee** (event span: *informed*). ACE does not record any event between these two entities because *informed* does not entail a CONTACT event for ACE event annotations. Another example that will clarify the difference is the following:

(2) In central Baghdad, [a Reuters cameraman] and [a cameraman for Spain's Telecinco] died when an American tank fired on the Palestine Hotel

ACE has annotated the above example as an event of type CONFLICT in which there are two entities that are of type person: the **Reuters cameraman** and the **cameraman for Spain's Telecinco**, both of which are arguments of type "Victim". Being an event that has two person entities involved makes the above sentence a potential social event. However, we do not record any event between these entities since the text does not reveal the cognitive states of the two entities; we do not know whether one was aware of the other.

ACE defines a class of social relations (PER-SOC) that records named relations like friendship, co-worker, long lasting etc. Also, there already exist systems that detect and classify these relations well. Therefore, even though these relations are directly relevant to our overall goal of social event extraction, we do not annotate, detect or classify these relations in this paper.

4 Tree Kernels, Discrete Structures, and Language

In this section, we give details of the structures and kernel we use for our classification tasks. We also discuss our motivation behind using these methods. Linear learning machines are one of the most popular machines used for classification problems. The objective of a typical classification problem is to learn a function that separates the data into different classes. The data is usually in the form of features extracted from abstract objects like strings, trees, etc. A drawback of learning by using complex functions is that complex functions do not generalize well and thus tend to over-fit. The research community therefore prefers linear classifiers over other complex classifiers. But more often than not, the data is not linearly separable. It can be made linearly separable by increasing the dimensionality of data but then learning suffers from the curse of dimensionality and classification becomes computationally intractable. This is where kernels come to the rescue. The well-known kernel trick aids us in finding similarity between feature vectors in a high dimensional space without having to write down the expanded feature space. The essence of kernel methods is that they compare two feature vectors in high dimensional space by using a dot product that is a function of the dot product of feature vectors in the lower dimensional space. Moreover, Convolution Kernels (first introduced by Haussler (1999)) can be used to compare abstract objects instead of feature vectors. This is because these kernels involve a recursive calculation over the "parts" of a discrete structure. This calculation is usually made computationally efficient using Dynamic Programming techniques. Therefore, Convolution Kernels alleviate the need of feature extraction (which usually requires domain knowledge, results in extraction of incomplete information and introduces noise in the data). Therefore, we use convolution kernels with a linear learning machine (Support Vector Machines) for our classification task.

Now we present the "discrete" structures followed by the kernel we used. We use the structures previously used by Nguyen et al. (2009), and propose one new structure. Although we experimented with all of their structures,[3] here we only present the ones that perform best for our classification task. All the structures and their combinations are derived from a variation of the underlying structures, Phrase Structure Trees (PST) and Dependency Trees (DT). For all trees we first extract their Path Enclosed Tree, which is the smallest common subtree that contains the two target entities (Moschitti, 2004). We use the Stanford parser (Klein and Manning, 2003) to get the basic PSTs and DTs. Following are the structures that we refer to in our experiments and results section:

PET: This refers to the smallest common phrase structure tree that contains the two target entities.

Dependency Words (DW) tree: This is the smallest common dependency tree that contains the two target entities. In Figure 1, since the target entities are at the leftmost and rightmost branch of the dependency tree, this is in fact a DW (ignoring the grammatical relations on the arcs).

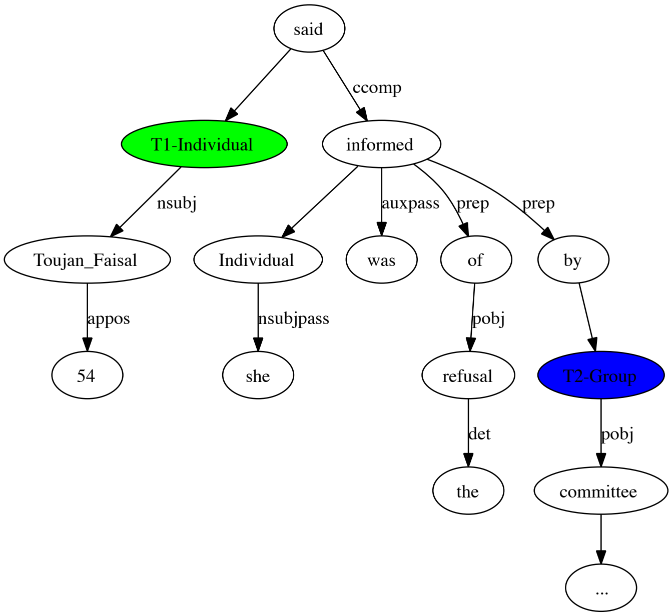


Figure 1: Dependency parse tree for the sentence (in the ACE corpus): "[Toujan Faisal], 54, {said} [she] was {informed} of the refusal by an [Interior Ministry committee] overseeing election preparations."

Grammatical Relation (GR) tree: If we replace the words at the nodes by their relation to their corresponding parent in DW, we get a GR tree. For example, in Figure 1, replacing *Toujan_Faisal* by *nsubj*, *54* by *appos*, *she* by *nsubjpass* and so on.

Grammatical Relation Word (GRW) tree: We get this tree by adding the grammatical relations as separate nodes between a node and its parent. For example, in Figure 1, adding *nsubj* as a node between *T1-Individual* and *Toujan_Faisal*, *appos* as a node between *54* and *Toujan_Faisal*, and so on.

Sequence Kernel of words (SK1): This is the sequence of words between the two entities, including their tags. For our example in Figure 1, it would be *T1-Individual Toujan Faisal 54 said she was informed of the refusal by an T2-Group Interior Ministry committee*.

Sequence in GRW tree (SqGRW): This is the new structure that we introduce which, to the best of

[3] We omitted SK6, which is the worst performing sequence kernel in (Nguyen et al., 2009).

our knowledge, has not been used before for similar tasks. It is the sequence of nodes from one target to the other in the GRW tree. For example, in Figure 1, this would be *Toujan_Faisal nsubj T1-Individual said ccomp informed prep by T2-Group pobj committee*.

We also use combinations of these structures (which we refer to as "combined-structures"). For example, PET_GR_SqGRW means we used the three structures (PET, GR and SqGRW) together with a kernel that calculates similarity between forests.

We use the Partial Tree (PT) kernel, first proposed by Moschitti (2006a), for structures derived from dependency trees and Subset Tree (SST) kernel, proposed by Collins and Duffy (2002), for structures derived from phrase structure trees. PT is a relaxed version of the SST; SST measures the similarity between two PSTs by counting all subtrees common to the two PSTs. However, there is one constraint: all daughter nodes of a node must be included. In PTs this constraint is removed. Therefore, in contrast to SSTs, PT kernels compare many more substructures. They have been used successfully by (Moschitti, 2004) for the task of semantic role labeling.

The choices we have made are motivated by the following considerations. We are interested in modeling classes of events which are characterized by the cognitive states of participants–who is aware of whom. The predicate-argument structure of verbs can encode much of this information very efficiently, and classes of verbs express their predicate-argument structure in similar ways. For example, many verbs of communication can express their arguments using the same pattern: *John talked/spoke/lectured/ranted/testified to Mary about Percy*. Independently of the verb, **John** is in a COG relation with **Percy** and in an INR relation with **Mary**. All these verbs allow us to drop either or both of the prepositional phrases, without altering the interpretation of the remaining constituents. And even more strikingly, any verb that can be put in that position is likely to have this interpretation; for example, we are likely to interpret the neologistic *John gazooked to Mary about Percy* as a similarly structured social event.

The regular relation between verb alternations and meaning components has been extensively studied (Levin, 1993; Schuler, 2005). This regularity in the syntactic predicate-argument structure allows us to overcome lexical sparseness. However, in order to exploit such regularities, we need to have access to a representation which makes the predicate-argument structure clear. Dependency representations do this. Phrase structure representations also represent predicate-argument structure, but in an indirect way through the structural configurations, and we expect this to increase the burden on the learner. (In some phrase structure representations, some arguments and adjuncts are not disambiguated.) When using dependency structures, the SST kernel is far less appealing, since it forces us to always consider all daughter nodes of a node. However, as we have seen, it is certain daughter nodes, such as the presence of a *to* PP and a *about* PP, which are important, while other daughters, such as temporal or locative adjuncts, should be disregarded. The PT kernel allows us to do this.

5 Sampling Methods

In this section we present the data sampling methods we use to deal with data skewness. We employ two well-known data sampling methods on the training data before creating a model for test data; random under-sampling and random over-sampling (Kotsiantis et al., 2006; Japkowicz, 2000; Weiss and Provost, 2001). These techniques are non-heuristic sampling methods that aim at balancing the class proportions by removing examples of the majority class and by duplicating instances of the minority class respectively. The reason for using these techniques is that learning is usually optimized to achieve high accuracy. Therefore, when presented with skewed training data, a classifier may learn the target concept with a high accuracy by only predicting the majority class. But if one looks at the precision, recall, and F-measure, of such a classifier, they will be very low for the minority class. Since, like other researchers, we are evaluating the goodness of a model based on its precision, recall and F-measure and not on the accuracy on the test set, either we should change the optimization function of the classifier or employ data sampling techniques. We employ the latter because by balancing the class ratio, we are presenting the classifier with a more challenging task of achieving a good accuracy when the majority base class is about 50%. The major drawbacks of the two techniques is that under-sampling throws away important information whereas over-sampling is prone to over-fitting (due to data duplication). As our results show, throwing away information about the majority class is much better than the system that tries to learn in an unbalanced scenario, but it performs worse than an approach using data duplication. Since we are using SVMs as a classifier, over-fitting is unlikely as reported by Kolcz et al. (2003).

In order to be sure that we are not over-fitting, we tried another sampling method proposed by Ha and Bunke (1997), which is shown to be good solution to avoid over-fitting by Chawla et al. (2002). This sampling technique proposes to generate synthetic examples of the minority class by "perturbing" the training data. Specifically, Ha and Bunke (1997) produced new synthetic examples for the task of handwritten character recognition by doing operations like rotation and skew on characters. The basic idea is to produce synthetic examples that are "close" to the real example from which these synthetic points are generated. Analogously, we tried two transformations on our dependency tree structures to produce synthetic examples. The first transformation is based on the observation that in control verb constructions, the matrix verb typically does not contribute to the interpretation as a social event or not. In this transformation, we lower the subject to an argument verb if it does not have a subject, and repeat this procedure iteratively. As it turned out, this transformation only occurred 15 times, and therefore it does not serve the purpose of over-sampling. We tried a more relaxed transformation on the rightmost target in the tree. Here, the observation is that for the COG social events, the second target may be very deeply embedded in the tree. For example, in Example 1, Toujan Faisal and the Interior Ministry Committee participate in a COG event (because Faisal is aware of the Committee during the saying event). However, the contents of what Faisal said is only relevant to the extent that it pertains to the committee. The depth of the embedding of the second target creates issues of data sparseness, as the path-enclosed trees become very large and very diverse. Our transformation, therefore, is to move the second target to its grandmother node, attaching it on the left, and to recalculate the path-enclosed tree, which is now smaller. This is repeated iteratively, so that a sentence with a deeply embedded second target can yield a large number of synthesized structures.

6 Experiments And Results

In this section we present experiments and results for our two tasks: social event detection and classification. For the social event detection task, we wish to validate the following research hypotheses. First, we aim to show the importance of using data sampling when evaluating on F-measure; specifically, we expect under-sampling to outperform no sampling, over-sampling to outperform under-sampling, and over-sampling with transformations to out perform over-sampling without transformations. In contrast, the social event classification task does not suffer from data skewness because the INR and COG relations; both occur almost the same number of times. Therefore, sampling methods may not be applied for this task. Second, for both tasks, we expect that a combination of kernels will out-perform individual kernels. Moreover, we expect that dependency trees will have a crucial role in achieving the best performance.

6.1 Experimental Set-up

We use part of ACE data that we annotated for social events. In all, we annotated 138 ACE documents. We retained the ACE entity annotations. We consider all entity mention pairs in a sentence. If our annotators recorded a relation between a pair of entity mentions, we say there is a relation between the corresponding entities. If there are any other pairs of entity mentions for the same pair of entity, we discard those. For all other pairs of entity mentions, we say there is no relation. Out of 138 files, four files did not have any positive or negative examples (because there were very few and sparse entity mentions in these four files). We found a total of 1291 negative examples, 172 examples belonging to class INR and 174 belonging to class COG.

We use Jet's sentence splitter[4] and the Stanford Parser (Klein and Manning, 2003) for phrase structure trees and dependency parses. For classifica-

[4] http://cs.nyu.edu/grishman/jet/jetDownload.html

tion, we used Alessandro Moschitti's SVM-Light-TK package (Moschitti, 2006b) which is built on the SVM-Light implementation of Joakhims (1999). For all our experiments, we perform 5-fold cross-validation. We randomly divide the whole corpus into 5 equal parts, such that no news story (or document) gets divided among two parts. For each fold, we then merge 4 parts to create a training corpus and treat the remaining part as a test corpus. By keeping individual news stories intact, we make sure that vocabulary specific to one story does not unrealistically improve the performance.

6.2 Social Event Detection

Social event detection is the task of detecting if any social event exists between a pair of entities in a sentence. We formulate the problem as a binary classification task by labeling an example that does not have a social event as class -1 and by labeling an example that either has an INR or COG social event as class 1. First we present results for our baseline system. Our baseline system uses various structures and their combinations but without any data balancing. [5]

Kernel	P	R	F1
PET	70.28	21.46	32.38
GR	**87.79**	15.21	25.55
GRW	76.42	8.26	14.8
SqGRW	48.78	6.08	10.38
PET_GR	70.21	**27.76**	**38.89**
PET_GR_SqGRW	71.06	26.74	38.02
GR_SqGRW	82.0	24.47	36.12
GRW_SqGRW	68.19	17.01	25.06
GR_GRW_SqGRW	79.81	21.99	32.57

Table 1: Baseline System for the task of social event detection. The proportion of positive data in training and test set is 21.1% and 20.6% respectively.

Table 1 presents results for our baseline system. Grammatical relation tree structure (GR), a structure derived from dependency tree by replacing the words by their grammatical relations achieves the best precision. This is probably because the classifier learns that if both the arguments of a predicate contain target entities then it is a social event. Among kernels for single structures, the path enclosed tree for PSTs (PET) achieves the best recall. Furthermore, a combination of structures derived from PSTs and DTs performs best. The sequence kernels, perform much worse than SqGRW (F1-measure as low as 0.45). Since it is the same case for all subsequent experiments, we omit them from the discussion.

Kernel	P	R	F1
PET	28.89	77.06	41.96
GR	**35.68**	72.47	47.37
GRW	29.7	83.6	43.6
SqGRW	34.31	**84.15**	**48.61**
PET_GR	34.38	83.94	**48.52**
PET_GR_SqGRW	34.34	83.66	**48.52**
GR_SqGRW	33.45	81.73	47.27
GRW_SqGRW	32.87	**84.44**	47.11
GR_GRW_SqGRW	32.73	83.26	46.82

Table 2: Under-sampled system for the task of relation detection. The proportion of positive examples in the training and test corpus is 50.0% and 20.6% respectively.

We now turn to experiments involving sampling. Table 2 presents results for under-sampling, i.e. randomly removing examples belonging to the negative class until its size matches the positive class. Table 2 shows a large gain in F1-measure of 9.72% absolute over the baseline system (Table 1). We found that worst performing kernel with under-sampling is SK1 with an F1-measure of 39.2% which is better than the best performance without under-sampling. These results make it clear that doing under-sampling greatly improves the performance of the classifier, despite the fact that we are using less training data (fewer negative examples). This is as expected because we are evaluating on F1-measure and the classifier is optimizing for accuracy.

Table 3 presents results for over-sampling i.e. replicating positive examples to achieve an equal number of examples belonging to the positive and negative class. Table 3 shows that the gain over the baseline system now is 22.2% absolute. Also, the gain over the under-sampled system is 12.5%

[5] Although we experimented with many more structures and their combinations, due to space restrictions we mention only the top results.

Kernel	P	R	F1
PET	50.9	57.21	53.62
GR	43.57	67.21	52.59
GRW	46.05	64.15	53.31
SqGRW	42.4	**72.75**	53.5
PET_GR	56.42	66.2	60.63
PET_GR_SqGRW	**57.28**	66.26	**61.11**
GR_SqGRW	44.35	71.17	54.52
GRW_SqGRW	44.77	68.79	54.12
GR_GRW_SqGRW	46.79	71.54	56.45

Table 3: Over-sampled system for the task of relation detection. The proportion of positive examples in the training and test corpus is 50.0% and 20.6% respectively.

absolute. As in the baseline system, a combination of structures performs best. As in the under-sampled system, when the data is balanced, SqGRW (sequence kernel on dependency tree in which grammatical relations are inserted as intermediate nodes) achieves the best recall. Here, the PET and GR kernel perform similar: this is different from the results of (Nguyen et al., 2009) where GR performed much worse than PET for ACE data. This exemplifies the difference in the nature of our event annotations from that of ACE relations. Since the average distance between target entities in the surface word order is higher for our events, the phrase structure trees are bigger. This means that implicit feature space is much sparser and thus not the best representation.

PET	37.04	66.49	47.28
GR	40.39	71.14	51.27
GRW	45.16	66.82	53.47
SqGRW	42.88	70.67	53.22
PET_GR	45.33	70.26	54.71
PET_GR_SqGRW	45.26	**72.97**	**55.67**
GR_SqGRW	43.73	71.47	54.06
GRW_SqGRW	45.70	71.30	55.32
GR_GRW_SqGRW	**45.91**	71.90	**55.70**

Table 4: Over-sampled System with transformation for relation detection. The proportion of positive examples in the training and test corpus is 51.7% and 20.6% respectively.

Table 4 presents results for using the over-sampling method with transformation that produces synthetic positive examples by using a transformation on dependency trees such that the new synthetic examples are “close” to the original examples. This method achieves a gain 16.78% over the baseline system. We expected this system to perform better than the over-sampled system but it does not. This suggests that our over-sampled system is not over-fitting; a concern with using oversampling techniques.

6.3 Social Event Classification

For the social event classification task, we only consider pairs of entities that have an event. Since these events could only be INR or COG, this is a binary classification problem. However, now we are interested in both outcomes of the classification, while earlier we were only interested in knowing how well we were finding relations (and not in how well we were finding “non-relations”). Therefore, accuracy is the relevant metric (Table 5).

Kernel	Acc
PET	76.85
GR	71.04
GRW	76.22
SqGRW	75.78
PET_GR	76.34
PET_GR_SqGRW	**78.72**
GR_SqGRW	75.60
GRW_SqGRW	76.96
GR_GRW_SqGRW	77.29

Table 5: System for the task of relation classification. The two classes are INR and COG, and we evaluate using accuracy (Acc.). The proportion of INR relations in training and test set is 49.7% and 49.63% respectively.

Even though the task of reasoning if an event is about one-way or mutual cognition seems hard, our system beats the chance baseline by 28.72%. These results show that there are significant clues in the lexical and syntactic structures that help in differentiating between interaction and cognition social events. Once again we notice that the combination of kernels works better than single kernels alone, though the difference here is less pronounced. Among the combined-structure approaches, combinations with dependency-derived structures continue to outperform those not including dependency (the best all-phrase structure performer is PET_SK1 with 75.7% accuracy, not shown in Table 5).

7 Conclusion And Future Work

In this paper, we have introduced the novel tasks of social event detection and classification. We show that data sampling techniques play a crucial role for the task of relation detection. Through oversampling we achieve an increase in F1-measure of 22.2% absolute over a baseline system. Our experiments show that as a result of how language expresses the relevant information, dependency-based structures are best suited for encoding this information. Furthermore, because of the complexity of the task, a combination of phrase based structures and dependency-based structures perform the best. This revalidates the observation of Nguyen et al. (2009) that phrase structure representations and dependency representations add complimentary value to the learning task. We also introduced a new sequence structure (SqGRW) which plays a role in achieving the best accuracy for both, social event detection and social event classification tasks.

In the future, we will use other parsers (such as semantic parsers) and explore new types of linguistically motivated structures and transformations. We will also investigate the relation between classes of social events and their syntactic realization.

Acknowledgments

The work was funded by NSF grant IIS-0713548. We thank Dr. Alessandro Moschitti and Truc-Vien T. Nguyen for helping us with re-implementing their system. We acknowledge Boyi Xie for his assistance in implementing the system. We would also like to thank Dr. Claire Monteleoni and Daniel Bauer for useful discussions and feedback.

References

Apoorv Agarwal, Owen Rambow, and Rebecca J Passonneau. 2010. Annotation scheme for social network extraction from text. In *Fourth Linguistic Annotation Workshop, ACL*.

N V Chawla, L O Hall, K W Bowyer, and W P Kegelmeyer. 2002. Smote: Synthetic minority oversampling technique. In *Journal of Artificial Intelligence Research*.

M. Collins and N. Duffy. 2002. Convolution kernels for natural language. In *Advances in neural information processing systems*.

Aron Culotta and Sorensen Jeffrey. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 423–429, Barcelona, Spain, July.

G Doddington, A Mitchell, M Przybocki, L Ramshaw, S Strassel, and R Weischedel. 2004. The automatic content extraction (ace) program–tasks, data, and evaluation. *LREC*, pages 837–840.

David K. Elson, Nicholas Dames, and Kathleen R. McKeown. 2010. Extracting social networks from literary fiction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*, Uppsala, Sweden.

Ralph Grishman, David Westbrook, and Adam Meyers Proc. 2005. Nyu's english ace 2005 system description. In *ACE Evaluation Workshop*.

Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. 2005. Exploring various knowledge in relation extraction. In *Proceedings of 43th Annual Meeting of the Association for Computational Linguistics*.

T. M. Ha and H Bunke. 1997. Off-line, handwritten numerical recognition by perturbation method. In *Pattern Analysis and Machine Intelligence*.

Sanda Harabagiu, Cosmin Adrian Bejan, and Paul Morarescu. 2005. Shallow semantics for relation extraction. In *International Joint Conference On Artificial Intelligence*.

David Haussler. 1999. Convolution kernels on discrete structures. Technical report, University of California at Santa Cruz.

Nathalie Japkowicz. 2000. Learning from imbalanced data sets: Comparison of various strategies. In *AAAI Workshop on Learning from Imbalanced Data Sets*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through unsupervised cross-document inference. In *Proceedings of ACL*.

Thorsten Joakhims. 1999. Making large-scale svm learning practical. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*.

Dan Klein and Chistopher D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *In Advances in Neural Information Processing Systems 15 (NIPS)*.

Aleksander Kolcz, Abdur Chowdhury, and Joshua Alspector. 2003. Data duplication: An imbalance problem. In *Workshop on Learning from Imbalanced Datasets, ICML*.

Sotiris Kotsiantis, Dimitris Kanellopoulos, and Panayiotis Pintelas. 2006. Handling imbalanced datasets: A review. In *GESTS International Transactions on Computer Science and Engineering*.

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

Alessandro Moschitti. 2004. A study on convolution kernels for shallow semantic parsing. In *Proceedings of the 42nd Conference on Association for Computational Linguistic*.

Alessandro Moschitti. 2006a. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proceedings of the 17th European Conference on Machine Learning*.

Alessandro Moschitti. 2006b. Making tree kernels practical for natural language learning. In *Proceedings of European chapter of Association for Computational Linguistics*.

Truc-Vien T. Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. 2009. Convolution kernels on constituent, dependency and sequential structures for relation extraction. *Conference on Empirical Methods in Natural Language Processing*.

Karin Kipper Schuler. 2005. *Verbnet: A Broad-Coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, upenncis.

Gary M Weiss and Foster Provost. 2001. The effect of class distribution on classifier learning: an empirical study. Technical Report ML.TR-44, Rutgers University, August.

D. Zelenko, C. Aone, and A. Richardella. 2002. Kernel methods for relation extraction. In *Proceedings of the EMNLP*.

Min Zhang, Jie Zhang, Jian Su, and Guodong Zhou. 2006. A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of COLING-ACL*.

Shubin Zhao and Ralph Grishman. 2005. Extracting relations with integrated information using kernel methods. In *Proceedings of the 43rd Meeting of the ACL*.

GuoDong Zhou, Min Zhang, DongHong Ji, and QiaoMing Zhu. 2007. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *Proceedings of EMNLP-CoNLL*.

Extracting Opinion Targets in a Single- and Cross-Domain Setting with Conditional Random Fields

Niklas Jakob
Technische Universität Darmstadt
Hochschulstraße 10
64289 Darmstadt, Germany

Iryna Gurevych
Technische Universität Darmstadt
Hochschulstraße 10
64289 Darmstadt, Germany

http://www.ukp.tu-darmstadt.de/people

Abstract

In this paper, we focus on the opinion target extraction as part of the opinion mining task. We model the problem as an information extraction task, which we address based on Conditional Random Fields (CRF). As a baseline we employ the supervised algorithm by Zhuang et al. (2006), which represents the state-of-the-art on the employed data. We evaluate the algorithms comprehensively on datasets from four different domains annotated with individual opinion target instances on a sentence level. Furthermore, we investigate the performance of our CRF-based approach and the baseline in a single- and cross-domain opinion target extraction setting. Our CRF-based approach improves the performance by 0.077, 0.126, 0.071 and 0.178 regarding F-Measure in the single-domain extraction in the four domains. In the cross-domain setting our approach improves the performance by 0.409, 0.242, 0.294 and 0.343 regarding F-Measure over the baseline.

1 Introduction

The automatic extraction and analysis of opinions has been approached on several levels of granularity throughout the last years. As opinion mining is typically an enabling technology for another task, this overlaying system defines requirements regarding the level of granularity. Some tasks only require an analysis of the opinions on a document or sentence level, while others require an extraction and analysis on a term or phrase level. Amongst the tasks which require the finest level of granularity are: a) Opinion question answering - i.e. with questions regarding an entity as in "What do the people like / dislike about X?". b) Recommender systems - i.e. if the system shall only recommend entities which have received good reviews regarding a certain aspect. c) Opinion summarization - i.e. if one wants to create an overview of all positive / negative opinions regarding aspect Y of entity X and cluster them accordingly. All of these tasks have in common that in order to fulfill them, the opinion mining system must be capable of identifying what the opinions in the individual sentences are about, hence extract the opinion targets.

Our goal in this work is to extract opinion targets from user-generated discourse, a discourse type which is quite frequently encountered today, due to the explosive growth of Web 2.0 community websites. Typical sentences which we encounter in this discourse type are shown in the following examples. The opinion targets which we aim to extract are underlined in the sentences, the corresponding opinion expressions are shown in italics.

(1) While none of the features are *earth-shattering*, eCircles does provide a *great* place to keep in touch.

(2) Hyundai's *more-than-modest* refresh has largely addressed all the original car's *weaknesses* while maintaining its price *competitiveness*.

The extraction of opinion targets can be considered as an instance of an information extraction (IE) task (Cowie and Lehnert, 1996). Conditional

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1035–1045,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

Random Fields (CRF) (Lafferty et al., 2001) have been successfully applied to several IE tasks in the past (Peng and McCallum, 2006). A recurring problem, which arises when working with supervised approaches, concerns the domain portability. In the opinion mining context this question has been prominently investigated with respect to opinion polarity analysis (sentiment analysis) in previous research (Aue and Gamon, 2005; Blitzer et al., 2007). Terms as “unpredictable” can express a positive opinion when uttered about the storyline of a movie but a negative opinion when the handling of a car is described. Hence the effects of training and testing a machine learning algorithm for sentiment analysis on data from different domains have been analyzed in previous research. However to the best of our knowledge, these effects have not been investigated regarding the extraction of opinion targets.

The contribution of this paper is a CRF-based approach for opinion targets extraction which tackles the problem of domain portability. We first evaluate our approach in three different domains against a state-of-the art baseline system and then evaluate the performance of both systems in a cross-domain setting. We show that our CRF-based approach outperforms the baseline in both settings, and how the diffrerent combinations of features we introduce influence the results of our CRF-based approach. The remainder of this paper is structured as follows: In Section 2 we discuss the related work, and in Section 3 we describe our CRF-based approach. Section 4 comprises our experimental setup including the description of the dataset we employ in our experiments in Section 4.1 and the baseline system in Section 4.2. The results of our experiments and their discussion follow in Section 5. Finally we draw our conclusions in Section 6.

2 Related Work

In the following we will discuss the related work regarding opinion target extraction and domain adaptation in opinion mining. The discussion of the related work on opinion target extraction is separated in supervised and unsupervised approaches. We conclude with a discussion of the related work on domain adaptation in opinion mining.

2.1 Unsupervised Opinion Target Extraction

The first work on opinion target extraction was done on customer reviews of consumer electronics. Hu and Liu (2004) introduce the task of *feature based summarization*, which aims at creating an overview of the product features commented on in the reviews. Their approach relies on a statistical analysis of the review terms based on association mining. A dataset of customer reviews from five domains was annotated by the authors regarding mentioned product features with respective opinion polarities. The association mining based algorithm yields a precision of 0.72 and a recall of 0.80 in the extraction of a manually selected subset of product features. The same dataset of product reviews was used in the work of Yi et al. (2003). They present and evaluate a complete system for opinion extraction which is based on a statistical analysis based on the Likelihood Ratio Test for opinion target extraction. The Likelihood Ratio Test yields a precision of 0.97 and 1.00 in the task of opinion target (product feature) extraction, recall values are not reported.

Popescu and Etzioni (2005) present the OPINE system for opinion mining on product reviews. Their algorithm is based on an information extraction system, which uses the pointwise mutual information based on the hitcounts of a web-search engine as an input. They evaluate the opinion target extraction separately on the dataset by Hu and Liu (2004). OPINE’s precision is on average 22% higher than the association mining based approach, while having an average 3% lower recall.

Bloom et al. (2007) manually create taxonomies of opinion targets for two datasets. With a handcrafted set of dependency tree paths their algorithm identifies related opinion expressions and targets. Due to the lack of a dataset annotated with opinion expressions and targets, they just evaluate the accuracy of several aspects of their algorithm by manually assessing an output sample. Their algorithm yields an accuracy of 0.75 in the identification of opinion targets.

Kim and Hovy (2006) aim at extracting opinion holders and opinion targets in newswire with semantic role labeling. They define a mapping of the semantic roles identified with FrameNet to the respective opinion elements. As a baseline, they implement an approach based on a dependency parser, which identifies the targets following the dependencies of opinion expressions. They measure the overlap between two human annotators and their algorithm as well as the baseline system. The algorithm based on semantic role labeling yields an F-Measure of 0.315 with annotator1 and 0.127 with annotator2, while the baseline yields an F-Measure of 0.107 and 0.109 regarding opinion target extraction

2.2 Supervised Opinion Target Extraction

Zhuang et al. (2006) present a supervised algorithm for the extraction of opinion expression - opinion target pairs. Their algorithm learns the opinion target candidates and a combination of dependency and part-of-speech paths connecting such pairs from an annotated dataset. They evaluate their system in a cross validation setup on a dataset of user-generated movie reviews and compare it to the results of the Hu and Liu (2004) system as a baseline. Thereby, the system by Zhuang et al. (2006) yields an F-Measure of 0.529 and outperforms the baseline which yields an F-Measure of 0.488 in the task of extracting opinion target - opinion expression pairs.

Kessler and Nicolov (2009) solely focus on identifying which opinion expression is linked to which opinion target in a sentence. They present a dataset of car and camera reviews in which opinion expressions and opinion targets are annotated. Starting with this information, they train a machine learning classifier for identifying related opinion expressions and targets. Their algorithm receives the opinion expression and opinion target annotations as input during runtime. The classifier is evaluated using the algorithm by Bloom et al. (2007) as a baseline. The support vector machine based approach by Kessler and Nicolov (2009) yields an F-Measure of 0.698, outperforming the baseline which yields an F-Measure of 0.445.

2.3 Domain Adaptation in Opinion Mining

The task of creating a supervised algorithm, which when trained on data from domain A, also performs well on data from another domain B, is a domain adaptation problem (Daumé III and Marcu, 2006; Jiang and Zhai, 2007). Aue and Gamon (2005) have investigated this challenge very early in the task of document level sentiment classification (positive / negative). They observe that increasing the amount of training data raises the classification accuracy, but only if the training data is from one source domain. Increasing the training data by mixing domains does not yield any consistent improvements. Blitzer et al. (2007) introduce an extension to a structural correspondence learning algorithm, which was specifically designed to address the task of domain adaptation. Their enhancement aims at identifying pivot features, which are stable across domains. In a series of experiments in document level sentiment classification they show that their extension outperforms the original structural correspondence learning approach. In their error analysis, the authors observe the best results were reached when the training - testing combinations were *Books - DVDs* or *Electronics - Kitchen appliances*. They conclude that the topical relatedness of the domains is an important factor. Furthermore they observe that training the algorithm on a smaller amount of data from a similar domain is more effective than increasing the amount of training data by mixing domains.

3 CRF-based Approach for Opinion Target Extraction

In the following we will describe the features we employ as input for our CRF-based approach. As the development data, we used 29 documents from the movies dataset, 23 documents from the web-services dataset and 15 documents from the cars & cameras datasets.

Token
This feature represents the string of the current token as a feature. Even though this feature is rather obvious, it can have considerable impact on the target extraction performance. If the vocabulary of targets is rather compact for a certain domain (corresponding to a low target type / target ratio), the training data is likely to contain the majority of the target types, which should hence be a good indicator. We will refer to this feature as **tk** in our result tables.

POS
This feature represents the part-of-speech tag of the current token as identified by the Stanford POS Tagger[1]. It can provide some means of lexical disam-

[1] http://nlp.stanford.edu/software/tagger.shtml

biguation, e.g. indicate that the token "sounds" is a noun and not a verb in a certain context. At the same time, the CRF algorithm is provided with additional information to extract opinion targets which are multiword expressions, i.e. noun combinations. We will refer to this feature as **pos** in our result tables.

Short Dependency Path
Previous research has successfully employed paths in the dependency parse tree to link opinion expressions and the corresponding targets (Zhuang et al., 2006; Kessler and Nicolov, 2009). Both works identify direct dependency relations such as "amod" and "nsubj" as the most frequent and at the same time highly accurate connections between a target and an opinion expression. We hence label all tokens which have a direct dependency relation to an opinion expression in a sentence. The Stanford Parser[2] is employed for the constituent and dependency parsing. We will refer to this feature as **dLn** in our result tables.

Word Distance
From the work of Zhuang et al. (2006) we can infer that opinion expressions and their target(s) are not always connected via short paths in the dependency parse tree. Since we cannot capture such paths with the abovementioned feature we introduce another feature which acts as heuristic for identifying the target to a given opinion expression. Hu and Liu (2004) and Yi et al. (2003) have shown that (base) noun phrases are good candidates for opinion targets in the datasets of product reviews. We therefore label the token(s) in the closest noun phrase regarding word distance to each opinion expression in a sentence. We will refer to this feature as **wrdDist** in our result tables.

Opinion Sentence
With this feature, we simply label all tokens occurring in a sentence containing an opinion expression. This feature shall enable the CRF algorithm to distinguish between the occurence of a certain token in a sentence which contains an opinion vs. a sentence without an opinion. We will refer to this feature as **sSn** in our result tables.

Our goal is to extract individual instances of opinion targets from sentences which contain an opinion expression. This can be modeled as a sequence segmentation and labeling task. The CRF algorithm receives a sequence of tokens $t_1 ... t_n$ for which it has to predict a sequence of labels $l_1 ... l_n$. We represent the possible labels following the IOB scheme: *B-Target*, identifying the beginning of an opinion target, *I-Target* identifying the continuation of a target, and *O* for other (non-target) tokens. We model the sentences as a linear chain CRF, which is based on an undirected graph. In the graph, each node corresponds to a token in the sentence and edges connect the adjacent tokens as they appear in the sentence. In our experiments, we use the CRF implementation from the Mallet toolkit[3].

4 Experimental Setup

4.1 Datasets

In our experiments, we employ datasets from three different sources, which span four domains in total (see Table 1). All of them consist of reviews collected from Web 2.0 sites. The first dataset consists of reviews for 20 different movies collected from the Internet Movie Database. It was presented in Zhuang et al. (2006) and annotated regarding opinion target - opinion expression pairs. The second dataset consists of 234 reviews for two different web-services collected from epinions.com, as described in Toprak et al. (2010). The third dataset is an extended version of the data presented in Kessler and Nicolov (2009). The authors have provided us with additional documents, which have been annotated in the meantime. The version of the dataset used in our experiments consists of 179 blog postings regarding different digital cameras and 336 reviews of different cars. In the description of their annotation guidelines, Kessler and Nicolov (2009) refer to opinion targets as mentions. Mentions are all aspects of the review topic, which can be targets of expressed opinions. However, not only mentions which occur as opinion targets were originally annotated, but also mentions which occur in non-opinion sentences. In our experiments, we only use the mentions which occur as targets of opinion expressions.

[2] http://nlp.stanford.edu/software/lex-parser.shtml

[3] http://mallet.cs.umass.edu/

All three datasets contain annotations regarding the antecedents of anaphoric opinion targets. In our experimental setup, we do not require the algorithms to also correctly resolve the antecedent of an opinion target representy by a pronoun, as we are solely interested in evaluating the opinion target extraction not any anaphora resolution.

As shown in rows 4 and 5 of Table 1, the documents from the cars and the cameras datasets exhibit a much higher density of opinions per document. 53.5% of the sentences from the cars dataset contain an opinion and in the cameras dataset even 56.1% of the sentences contain an opinion, while in the movies and the web-services reviews just 22.1% and 22.4% of the sentences contain an opinion. Furthermore in the cars and the cameras datasets the lexical variability regarding the opinion targets is substantially larger than in the other two datasets: We calculate *target types* by counting the number of distinct opinion targets in a dataset. We divide this by the sum of all opinion target instances in the dataset. For the cars dataset this ratio is 0.440 and for the cameras dataset it is 0.433, while for the web-services dataset it is 0.306 and for the movies dataset only 0.122. In terms of reviews this means, that in the movie reviews the same movie aspects are repeatedly commented on, while in the cars and the cameras datasets many different aspects of these entities are discussed, which in turn each occur infrequently.

Table 1: Dataset Statistics

	movies	**web-services**	**cars**	**cameras**
Documents	1829	234	336	179
Sentences	24555	6091	10969	5261
Tokens / sentence	20.3	17.5	20.3	20.4
Sentences with target(s)	21.4%	22.4%	51.1%	54.0%
Sentences with opinion(s)	21.4%	22.4%	53.5%	56.1%
Targets	7045	1875	8451	4369
Target types	865	574	3722	1893
Tokens / target	1.21	1.35	1.29	1.42
Avg. targets / opinion sent.	1.33	1.37	1.51	1.53

4.2 Baseline System

In the task of opinion target extraction the supervised algorithm by Zhuang et al. (2006) represents the state-of-the-art on the movies dataset we also employ in our experiments. We therefore use it as a baseline. The algorithm learns two aspects from the labeled training data:

1. A set of opinion target candidates
2. A set of paths in a dependency tree which identify valid opinion target - opinion expression pairs

In our experiments, we learn the full set of opinion targets from the labeled training data in the first step. This is slightly different from the approach in (Zhuang et al., 2006), but we expect that this modification should be beneficial for the overall performance in terms of recall, as we do not remove any learned opinion targets from the candidate list. In the second step, the annotated sentences are parsed and a graph containing the words of a sentence is created, which are connected by the dependency relations between them. For each opinion target - opinion expression pair from the gold standard, the shortest path connecting them is extracted from the dependency graph. A path consists of the part-of-speech tags of the nodes and the dependency types of the edges. Example 3 shows a typical dependency path.

(3) NN - nsubj - NP - amod - JJ

During runtime, the algorithm identifies opinion targets from the candidate list in the training data. The opinion expressions are directly taken from the gold standard, as we focus on the opinion target extraction aspect in this work. The sentences are then parsed and if a valid path between a target and an opinion expression is found in the list of possible paths, then the pair is extracted. Since the dependency paths only identify pairs of single word target and opinion expression candidates, we employ a merging step. Extracted target candidates are merged into a multiword target if they are adjacent in a sentence. Thereby, the baseline system is also capable of extracting multiword opinion targets.

4.3 Metrics

We employ the following requirements in our evaluation of the opinion target extraction: An opinion target must be extracted with exactly the span boundaries as annotated in the gold standard. This is especially important regarding multiword targets. Extracted targets which partially overlap with the annotated gold standard are counted as errors. Hence a target extracted by the algorithm which does not exactly match the boundaries of a target in the gold standard is counted as a false positive (FP), e.g. if "battery life" is annotated as the target in the gold standard, only "battery" or "life" extracted as targets will be counted as FPs. Exact matches between the targets extracted by the algorithm and the gold standard are true positives (TP). We refer to the number of annotated targets in the gold standard as T_{GS}. Precision is calculated as $Precision = \frac{TP}{TP+FP}$, and recall is calculated as $Recall = \frac{TP}{T_{GS}}$. F-Measure is the harmonic mean of precision and recall.

5 Results and Discussion

We investigate the performance of the baseline and the CRF-based approach for opinion target extraction in a single- and cross-domain setting. The single-domain approach assumes that there is a set of training data available for the same domain as the domain the algorithm is being tested on. In this setup, we will both run the baseline and our CRF based system in a 10-fold cross-validation and report results macro averaged over all runs. In the cross-domain approach, we will investigate how the algorithm performs if given training data from domain A while being tested on another domain B. In this setting, we will train the algorithm on the entire dataset A, and test it on the entire dataset B, we hence report one micro averaged result set. In Subsection 5.1 we present the results of both the baseline system and our CRF-based approach in the single-domain setting, in Subsection 5.2 we present the results of the two systems in the cross-domain opinion target extraction.

Table 2: Single-Domain Extraction with Zhuang Baseline

Dataset	Precision	Recall	F-Measure
movies	0.663	0.592	0.625
web-services	0.624	0.394	0.483
cars	0.259	0.426	0.322
cameras	0.423	0.431	0.426

5.1 Single-Domain Results

5.1.1 Zhuang Baseline

As shown in Table 2, the state-of-the-art algorithm of Zhuang et al. (2006) performs best on the movie review dataset and worst on the cars dataset. The results on the movie dataset are higher than originally reported in (Zhuang et al., 2006) (Precision 0.483, Recall 0.585, F-Measure 0.529). We assume that this is due to two reasons: 1. In our task, the algorithm uses the opinion expression annotation from the gold standard. 2. We do not remove any learned opinion target candidates from the training data (See Section 4.2).

During training we observed that for each dataset the lists of possible dependency paths (see Example 3) contained several hundred entries, many of them only occurring once. We assume that the recall of the algorithm is limited by a large variety of possible dependency paths between opinion targets and opinion expressions, since the algorithm cannot link targets and opinion expressions in the testing data if there is no valid candidate dependency path. Furthermore, we observe that for the cars dataset the size of the dependency path candidate list (6642 entries) was approximately five times larger than the dependency graph candidate list for the web-services dataset (1237 entries), which has a comparable size regarding documents. At the same time, the list of target candidates of the cars dataset was approximately eight times larger than the target candidate list for the web-services dataset. We assume that a large number of both the target candidates as well as the dependency path candidates introduces many false positives during the target extraction, hence lowering the precision of the algorithm on the cars dataset considerably.

Table 3: Single-Domain Extraction with our CRF-based Approach

	movies			**web-services**			**cars**			**cameras**		
Features	**Prec**	**Rec**	**F-Me**	**Prec**	**Rec**	**F-Me**	**Prec**	**Rec**	**F-Me**	**Prec**	**Rec**	**F-Me**
tk, pos	0.639	0.133	0.220	0.500	0.051	0.093	0.438	0.110	0.175	0.300	0.085	0.127
tk, pos, wDs	0.542	0.181	0.271	0.451	0.272	0.339	0.570	0.354	0.436	0.549	0.375	0.446
tk, pos, dLn	0.777	0.481	0.595	0.634	0.380	0.475	0.603	0.372	0.460	0.569	0.376	0.453
tk, pos, sSn	0.673	0.637	0.653	0.604	0.397	0.476	0.453	0.180	0.257	0.398	0.172	0.238
tk, pos, dLn, wDs	**0.792**	0.481	0.598	0.620	0.354	0.450	0.603	0.389	0.473	0.596	0.425	0.496
tk, pos, sSn, wDs	0.662	0.656	0.659	0.664	0.461	0.544	0.564	0.370	0.446	0.544	0.381	0.447
tk, pos, sSn, dLn	0.791	0.477	0.594	0.654	0.501	0.568	0.598	0.384	0.467	0.586	0.391	0.468
tk, pos, sSn, dLn, wDs	0.749	**0.661**	**0.702**	**0.722**	**0.526**	**0.609**	**0.622**	**0.414**	**0.497**	0.614	**0.423**	**0.500**
pos, sSn, dLn, wDs	0.672	0.441	0.532	0.612	0.322	0.422	0.612	0.369	0.460	**0.674**	0.398	0.500

5.1.2 Our CRF-based Approach

Table 3 shows the results of the opinion target extraction using the CRF algorithm. Row 8 contains the results of the feature configuration, which yields the best performance regarding F-Measure across all datasets. We observe that our aproach outperforms the Zhuang baseline on all datasets. The gain in F-Measure is between 0.077 in the movies domain and 0.175 in the cars domain. Although the CRF-based approach clearly outperforms the baseline system on all four datasets, we also observe the same general trend regarding the individual results: The CRF yields the best results on the movies dataset and the worst results on the cars & cameras dataset.

As shown in the first row, the results when using just the token string and part-of-speech tags as features are very low, especially regarding recall. We observe that the higher the lexical variability of the opinion targets is in a dataset, the lower the results are. If we add the feature based on word distance (row 2), the recall is improved on all datasets, while the precision is slightly lowered on the movies and web-services datasets. The dependency path based feature performs better compared to the word distance heuristic as shown in row 3. The precision is considerably increased on all datasets, on the movies and cars & cameras datasets even reaching the overall highest value. At the same time, we observe an increase of recall on all datasets. The observation made in previous research that short paths in the dependency graph are a high precision indicator of related opinion expressions - opinion targets (Kessler and Nicolov, 2009) is confirmed on all datasets. Adding the information regarding opinion sentences to the basic features of the token string and the part-of-speech tag (row 4) yields the biggest improvements regarding F-Measure on the movies and web-services dataset (+0.433 / +0.383). On the cars & cameras dataset the recall is relatively low again. We assume that this is again due to the high lexical variability, so that the CRF algorithm will encounter many actual opinion targets in the testing data which have not occurred in the training data and will hence not be extracted.

As shown in row 5, if we combine the dependency graph based feature with the word distance heuristic, the results regarding F-Measure are consistently higher than the results of these features in isolation (rows 2 - 4) on all datasets. We conclude that these two features are complementary, as they apparently indicate different kinds of opinion targets which are then correctly extracted by the CRF. If we combine each of the opinion expression related features with the label which identifies opinion sentences in general (rows 6 & 7), we observe that this feature is also complementary to the others. On all datasets the results regarding F-Measure are consistently higher compared to the features in isolation (rows 2 - 4). Row 8 shows the results of all features in combination. Again, we observe the complementarity of the features, as the results of this feature combination are the best regarding F-Measure across all datasets.

In row 9 of the results, we exclude the token string as a feature. In comparison to the full feature combination of row 8 we observe a significant decrease of F-Measure on the movies and the web-services dataset. On the cars dataset we only observe a slight decrease of recall. Interestingly on the cameras dataset we even observe a slight increase of precision which compensates a slight decrease of recall, in turn resulting in stable F-Measure of 0.500 as in the full feature set of row 8.

We have run some additional experiments in which we did not rely on the annotated opinion expressions, but employed a general pupose subjectivity lexicon[4]. Already in the single-domain extraction, we observed that the results declined substantially (e.g. web-services F-Measure: 0.243, movies F-Measure: 0.309, cars F-Measure: 0.192 and cameras F-Measure: 0.198).

We performed a quantitative error analysis on the results of the CRF-based approach in the single-domain setting. In doing so, we focused on misclassifications of B-Target and I-Target instances, as the recall is consistently lower than the precision across all datasets. We observe that most of the recall errors result from one-word opinion targets or the beginning of opinion targets (B-Targets) being missclassified as non-targets (movies 83%, web-services 73%, cars 68%, cameras 64%). For the majority of these missclassifications neither the *dLn* nor the *wDs* features were present (movies 82%, web-services 56%, cars 64%, cameras 61%). We assume that our features cannot capture the structure of more complex sentences very well. Our results indicate that the *dLn* and *wDs* features are complementary, but apparently there are quite a few cases in which the opinion target is neither directly related to the opinion expression in the dependency graph nor close to it in the sentence. One of these sentences, in this case from a camera review, in shown in Example 4.

(4) A lens cap and a strap may not sound very important, but it makes a *huge difference* in the speed and usability of the camera.

In this sentence, the *dLn* and *wDs* features both labeled “speed” which was incorrectly extracted as the target of the opinion. None of the actual targets “lens cap”, “strap” and “camera” have a short dependency path to the opinion expression and “speed” is simply the closest noun to it. Note that although both “speed” and “usability” are attributes of a camera, the opinion in this sentence is about the “lens cap” and “strap”, hence only these attributes are annotated as targets.

5.2 Cross-Domain Results

5.2.1 Zhuang Baseline

Table 4 shows the results of the opinion target extraction with the state-of-the-art system in the cross-domain setting. We observe that the results on all domain combinations are very low. A quantitative error analysis has revealed that there is hardly any overlap in the opinion target candidates between domains, as reflected by the low recall in all configurations. The vocabularies of the opinion targets are too different, hence the performance of the algorithm by Zhuang et al. (2006) is so low. The overlap regarding the dependency paths between domains was however higher. Especially identical short paths could be found across domains which at the same time typically occured quite often. For future work it might be interesting to investigate how the algorithm by Zhuang et al. (2006) performs in the cross-domain setting if the target candidate learning is performed differently, e.g. with a statistical approach as outlined in Section 2.1.

5.2.2 CRF-based Approach

The results of the cross-domain target extraction with the CRF-based algorithm are shown in Table 5. Due to the increase of system configurations introduced by the training - testing data combinations, we had to limit results of the feature combinations reported in the Table. The feature combination *pos, sSn, wDs, dLn* yielded the best results regarding F-Measure. Hence, we report its result as the basic feature set. When comparing the results of the best performing feature / training data combination of our CRF-based approach with the baseline, we observe that our approach outperforms the baseline on all four domains. The gain in F-Measure is 0.409 in the movies domain, 0.242 in the web-services domain, 0.294 in the cars domain and 0.343 in the cameras domain.

Effects of Features

Interestingly with the best performing feature combination from the single-domain extraction, the results regarding recall in the cross-domain extraction are very low[5]. This is due to the fact that the CRF attributed a relatively large weight to the token string

[4] http://www.cs.pitt.edu/mpqa/

[5] Not shown in any result table due to limited space.

Table 4: Cross-Domain Extraction with Zhuang Baseline

Training	Testing	Precision	Recall	F-Measure
web-services	movies	**0.194**	0.032	0.055
cars	movies	0.032	0.034	0.033
cameras	movies	0.155	0.084	**0.109**
cars + cameras	movies	0.071	0.104	0.084
web-services + cars + cameras	movies	0.070	0.103	0.083
movies	web-services	**0.311**	0.073	**0.118**
cars	web-services	0.086	0.091	0.089
cameras	web-services	0.164	0.081	0.108
cars + cameras	web-services	0.086	**0.104**	0.094
movies + cars + cameras	web-services	0.074	0.100	0.080
movies	cars	0.182	0.014	0.026
web-services	cars	0.218	0.028	0.049
cameras	cars	**0.250**	0.121	0.163
cameras + web-services	cars	0.247	**0.131**	**0.171**
movies + web-services	cars	0.246	0.045	0.076
movies	cameras	0.108	0.012	0.022
web-services	cameras	**0.268**	0.048	0.082
cars	cameras	0.125	**0.160**	**0.140**
cars + web-services	cameras	0.119	0.157	0.136
movies + web-services	cameras	0.245	0.063	0.100

feature. As we also observed in the analysis of the baseline results, the overlap of the opinion target vocabularies between domains is low, which resulted in a very small number of targets extracted by the CRF. As shown in Table 5 the results are promising regarding F-Measure if we just leave the token feature out of the configuration.

Effects of Training Data

When analyzing the results of the different training - testing domain configurations we observe the following: In isolation training data from the cameras domain consistently yields the best results regarding F-Measure when the algorithm is run on the datasets from the other three domains. This is particularly interesting since the cameras dataset is the smallest of the four (see Table 1). We investigated whether the CRF algorithm was overfitting to the training datasets by reducing their size to the size of the cameras dataset. However, the reduction of the training data sizes never improved the extraction results regarding F-Measure for the movies, web-serviecs and cars datasets. The good results when training on the cameras dataset are in line with our observations from Section 5.1.2. We noticed that on the cameras dataset the results regarding F-Measure remained stable if the token feature is not used in the training.

In isolation, training only on the cars data yields the second highest results on the movies and web-services datasets and the highest results regarding F-Measure on the cameras data. However, the results of the cars + cameras training data combination indicate that the cameras data does not contribute any additional information during the learning, since the results on both the movies and the web-services datasets are lower than when training only on the cameras data.

Our results also confirm the insights gained by Blitzer et al. (2007), who observed that in cross-domain polarity analysis adding more training data is not always beneficial. Apparently even the smallest training dataset (cameras) contain enough feature instances to learn a model which performs well on the testing data.

We observe that the results of the cross-domain extraction regarding F-Measure come relatively close to the results of the single-domain setting, especially if the token string feature is removed there (see Table 3 row 9). On the cars and the cameras dataset the cross-domain results are even closer to the single-domain results. The features we employ seem to scale well across domains and compensate the difference between training and testing data and the lack of information regarding the target vocabu-

Table 5: Cross-Domain Extraction with our CRF-based Approach

		Testing					
		web-services			movies		
		Pre	Rec	F-Me	Pre	Rec	F-Me
Training	web-services	-	-	-	0.560	0.339	0.422
	movies	**0.565**	0.219	0.316	-	-	-
	cars	0.538	0.248	0.340	0.642	0.382	0.479
	cameras	0.529	0.256	0.345	0.642	0.408	0.499
	movies + cars	0.554	0.249	0.344	-	-	-
	movies + cameras	0.530	**0.273**	**0.360**	-	-	-
	movies + cars + cameras	0.562	0.250	0.346	-	-	-
	cars + cameras	0.538	0.254	0.345	0.641	0.395	0.489
	web-services + cars	-	-	-	**0.651**	0.396	0.492
	web-services + cameras	-	-	-	0.642	**0.435**	**0.518**
	web-services + cars + cameras	-	-	-	0.639	0.405	0.496
		cars			cameras		
		Pre	Rec	F-Me	Pre	Rec	F-Me
	web-services	0.391	0.277	0.324	0.505	0.330	0.399
	movies	0.512	0.307	0.384	0.550	0.303	0.391
	cars	-	-	-	0.665	0.369	0.475
	cameras	**0.589**	0.384	**0.465**	-	-	-
	cameras + movies	0.567	**0.394**	**0.465**	-	-	-
	cameras + web-services	0.572	0.381	0.457	-	-	-
	movies + web-services	0.489	0.327	0.392	0.553	0.339	0.421
	movies + cars	-	-	-	0.634	0.376	0.472
	web-services + cars	-	-	-	**0.678**	0.376	**0.483**
	web-services + movies + cars	-	-	-	0.635	**0.378**	0.474
	movies + web-services + cameras	0.549	0.381	0.450	-	-	-

lary.

6 Conclusions

In this paper, we have shown how a CRF-based approach for opinion target extraction performs in a single- and cross-domain setting. We have presented a comparative evaluation of our approach on datasets from four different domains. In the single-domain setting, our CRF-based approach outperforms a supervised baseline on all four datasets. Our error analysis indicates that additional features, which can capture opinions in more complex sentences, are required to improve the performance of the opinion target extraction. Our CRF-based approach also yields promising results in the cross-domain setting. The features we employ scale well across domains, given that the opinion target vocabularies are substantially different. For future work, we might investigate how machine learning algorithms, which are specifically designed for the problem of domain adaptation (Blitzer et al., 2007; Jiang and Zhai, 2007), perform in comparison to our approach. Since three of the features we employed in our CRF-based approach are based on the respective opinion expressions, it is to investigate how to mitigate the possible negative effects introduced by errors in the opinion expression identification if they are not annotated in the gold standard. We observe similar challenges as Choi et al. (2005) regarding the analysis of complex sentences. Although our data is user-generated from Web 2.0 communities, a manual inspection has shown that the documents were of relatively high textual quality. It is to investigate to which extent the approaches taken in the analysis of newswire, such as identifying targets with coreference resolution, can also be applied to our task on user-generated discourse.

Acknowledgments

The project was funded by means of the German Federal Ministry of Economy and Technology under the promotional reference “01MQ07012”. The authors take the responsibility for the contents. This work has been supported by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant No. I/82806.

References

Anthony Aue and Michael Gamon. 2005. Customizing sentiment classifiers to new domains: A case study. In *Proceedings of the 5th International Conference on Recent Advances in Natural Language Processing*, Borovets, Bulgaria, September.

John Blitzer, Mark Dredze, 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 of Computational Linguistics*, pages 440–447, Prague, Czech Republic, June.

Kenneth Bloom, Navendu Garg, and Shlomo Argamon. 2007. Extracting appraisal expressions. In *Proceedings of Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*, pages 308–315, Rochester, New York, USA, April.

Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Identifying sources of opinions with conditional random fields and extraction patterns. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 355–362, Vancouver, Canada, October.

James R. Cowie and Wendy G. Lehnert. 1996. Information extraction. *Communications of the ACM*, 39(1):80–91.

Hal Daumé III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research (JAIR)*, 26:101–126.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 168–177, Seattle, Washington, USA, August.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 264–271, Prague, Czech Republic, June. Association for Computational Linguistics.

Jason Kessler and Nicolas Nicolov. 2009. Targeting sentiment expressions through supervised ranking of linguistic configurations. In *Proceedings of the Third International AAAI Conference on Weblogs and Social Media*, pages 90–97, San Jose, California, USA, May.

Soo-Min Kim and Eduard Hovy. 2006. Extracting opinions, opinion holders, and topics expressed in online news media text. In *Proceedings of the ACL Workshop on Sentiment and Subjectivity in Text*, pages 1–8, Sydney, Australia, July.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, Williamstown, MA, USA, June.

Fuchun Peng and Andrew McCallum. 2006. Information extraction from research papers using conditional random fields. *Information Processing and Management*, 42(4):963–979, July.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 339–346, Vancouver, Canada, October.

Cigdem Toprak, Niklas Jakob, and Iryna Gurevych. 2010. Sentence and expression level annotation of opinions in user-generated discourse. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 575–584, Uppsala, Sweden, July.

Jeonghee Yi, Tetsuya Nasukawa, Razvan Bunescu, and Wayne Niblack. 2003. Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 427–434, Melbourne, Florida, USA, December.

Li Zhuang, Feng Jing, and Xiao-Yan Zhu. 2006. Movie review mining and summarization. In *Proceedings of the ACM 15th Conference on Information and Knowledge Management*, pages 43–50, Arlington, Virginia, USA, November.

Multi-level Structured Models for Document-level Sentiment Classification

Ainur Yessenalina
Dept. of Computer Science
Cornell University
Ithaca, NY, USA
ainur@cs.cornell.edu

Yisong Yue
Dept. of Computer Science
Cornell University
Ithaca, NY, USA
yyue@cs.cornell.edu

Claire Cardie
Dept. of Computer Science
Cornell University
Ithaca, NY, USA
cardie@cs.cornell.edu

Abstract

In this paper, we investigate structured models for document-level sentiment classification. When predicting the sentiment of a subjective document (e.g., as positive or negative), it is well known that not all sentences are equally discriminative or informative. But identifying the useful sentences automatically is itself a difficult learning problem. This paper proposes a joint two-level approach for document-level sentiment classification that simultaneously extracts useful (i.e., subjective) sentences and predicts document-level sentiment based on the extracted sentences. Unlike previous joint learning methods for the task, our approach (1) does not rely on gold standard sentence-level subjectivity annotations (which may be expensive to obtain), and (2) optimizes directly for document-level performance. Empirical evaluations on movie reviews and U.S. Congressional floor debates show improved performance over previous approaches.

1 Introduction

Sentiment classification is a well-studied and active research area (Pang and Lee, 2008). One of the main challenges for document-level sentiment categorization is that not every part of the document is equally informative for inferring the sentiment of the whole document. Objective statements interleaved with the subjective statements can be confusing for learning methods, and subjective statements with conflicting sentiment further complicate the document categorization task. For example, authors of movie reviews often devote large sections to (largely objective) descriptions of the plot (Pang and Lee, 2004). In addition, an overall positive review might still include some negative opinions about an actor or the plot.

Early research on document-level sentiment classification employed conventional machine learning techniques for text categorization (Pang et al., 2002). These methods, however, assume that documents are represented via a flat feature vector (e.g., a bag-of-words). As a result, their ability to identify and exploit subjectivity (or other useful) information at the sentence-level is limited.

And although researchers subsequently proposed methods for incorporating sentence-level subjectivity information, existing techniques have some undesirable properties. First, they typically require gold standard sentence-level annotations (McDonald et al. (2007), Mao and Lebanon (2006)). But the cost of acquiring such labels can be prohibitive. Second, some solutions for incorporating sentence-level information lack mechanisms for controlling how errors propagate from the subjective sentence identification subtask to the main document classification task (Pang and Lee, 2004). Finally, solutions that attempt to handle the error propagation problem have done so by explicitly optimizing for the best *combination* of document- and sentence-level classification accuracy (McDonald et al., 2007). Optimizing for this compromise, when the real goal is to maximize only the document-level accuracy, can potentially hurt document-level performance.

In this paper, we propose a joint two-level model to address the aforementioned concerns. We formulate our training objective to directly optimize for

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1046–1056,
MIT, Massachusetts, USA, 9-11 October 2010.

document-level accuracy. Further, we do not require gold standard sentence-level labels for training. Instead, our training method treats sentence-level labels as hidden variables and *jointly learns* to predict the document label and those (subjective) sentences that best "explain" it, thus controlling the propagation of incorrect sentence labels. And by directly optimizing for document-level accuracy, our model learns to solve the sentence extraction subtask only to the extent required for accurately classifying document sentiment. A software implementation of our method is also publicly available.[1]

For the rest of the paper, we will discuss related work, motivate and describe our model, present an empirical evaluation on movie reviews and U.S. Congressional floor debates datasets and close with discussion and conclusions.

2 Related Work

Pang and Lee (2004) first showed that sentence-level extraction can improve document-level performance. They used a cascaded approach by first filtering out objective sentences and performing subjectivity extractions using a global min-cut inference. Afterward, the subjective extracts were converted into inputs for the document-level sentiment classifier. One advantage of their approach is that it avoids the need for explicit subjectivity annotations. However, like other cascaded approaches (e.g., Thomas et al. (2006), Mao and Lebanon (2006)), it can be difficult to control how errors propagate from the sentence-level subtask to the main document classification task.

Instead of taking a cascaded approach, one can directly modify the training of flat document classifiers using lower level information. For instance, Zaidan et al. (2007) used human annotators to mark the "annotator rationales", which are text spans that support the document's sentiment label. These annotator rationales are then used to formulate additional constraints during SVM training to ensure that the resulting document classifier is less confident in classifying a document that does not contain the rationale versus the original document. Yessenalina et al. (2010) extended this approach to use automatically generated rationales.

A natural approach to avoid the pitfalls associated with cascaded methods is to use joint two-level models that simultaneously solve the sentence-level and document-level tasks (e.g., McDonald et al. (2007), Zaidan and Eisner (2008)). Since these models are trained jointly, the sentence-level predictions affect the document-level predictions and vice-versa. However, such approaches typically require sentence-level annotations during training, which can be expensive to acquire. Furthermore, the training objectives are usually formulated as a compromise between sentence-level and document-level performance. If the goal is to predict well at the document-level, then these approaches are solving a much harder problem that is not exactly aligned with maximizing document-level accuracy.

Recently, researchers within both Natural Language Processing (e.g., Petrov and Klein (2007), Chang et al. (2010), Clarke et al. (2010)) and other fields (e.g., Felzenszwalb et al. (2008), Yu and Joachims (2009)) have analyzed joint multi-level models (i.e., models that simultaneously solve the main prediction task along with important subtasks) that are trained using limited or no explicit lower level annotations. Similar to our approach, the lower level labels are treated as hidden or latent variables during training. Although the training process is non-trivial (and in particular requires a good initialization of the hidden variables), it avoids the need for human annotations for the lower level subtasks. Some researchers have also recently applied hidden variable models to sentiment analysis, but they were focused on classifying either phrase-level (Choi and Cardie, 2008) or sentence-level polarity (Nakagawa et al., 2010).

3 Extracting Hidden Explanations

In this paper, we take the view that each document has a subset of sentences that best explains its sentiment. Consider the "annotator rationales" generated by human judges for the movie reviews dataset (Zaidan et al., 2007). Each rationale is a text span that was identified to support (or explain) its parent document's sentiment. Thus, these rationales can be interpreted as (something close to) a ground truth labeling of the explanatory segments. Using a dataset where each document contains only its rationales,

[1] http://projects.yisongyue.com/svmsle/

Algorithm 1 Inference Algorithm for (2)

```
Input: $x$
Output: $(y, s)$
$s_+ \leftarrow \mathrm{argmax}_{s \in S(x)} \vec{w}^T \Psi(x, +1, s)$
$s_- \leftarrow \mathrm{argmax}_{s \in S(x)} \vec{w}^T \Psi(x, -1, s)$
if $\vec{w}^T \Psi(x, +1, s_+) > \vec{w}^T \Psi(x, -1, s_-)$ then
    Return $(+1, s_+)$
else
    Return $(-1, s_-)$
end if
```

cross validation experiments using an SVM classifier yields 97.44% accuracy – as opposed to 86.33% accuracy when using the full text of the original documents. Clearly, extracting the best supporting segments can offer a tremendous performance boost.

We are interested in settings where human-extracted explanations such as annotator rationales might not be readily available, or are imperfect. As such, we will formulate the set of extracted sentences as latent or hidden variables in our model. Viewing the extracted sentences as latent variables will pose no new challenges during prediction, since the model is expected to predict all labels at test time. We will leverage recent advances in training latent variable SVMs (Yu and Joachims, 2009) to arrive at an effective training procedure.

4 Model

In this section, we present a two-level document classification model. Although our model makes predictions at both the document and sentence levels, it will be trained (and evaluated) only with respect to document-level performance. We begin by presenting the feature structure and inference method. We will then describe a supervised training algorithm based on structural SVMs, and finally discuss some extensions and design decisions.

Let x denote a document, $y = \pm 1$ denote the sentiment (for us, a binary positive or negative polarity) of a document, and s denote a subset of explanatory sentences in x. Let $\Psi(x, y, s)$ denote a joint feature map that outputs features describing the quality of predicting sentiment y using explanation s for document x. We focus on linear models, so given a (learned) weight vector $\vec{w}$, we can write the quality of predicting y (with explanation s) as

$$F(x, y, s; \vec{w}) = \vec{w}^T \Psi(x, y, s), \tag{1}$$

and a document-level sentiment classifier as

$$h(x; \vec{w}) = \underset{y=\pm 1}{\mathrm{argmax}} \max_{s \in S(x)} F(x, y, s; \vec{w}), \tag{2}$$

where $S(x)$ denotes the collection of feasible explanations (e.g., subsets of sentences) for x.

Let x^j denote the j-th sentence of x. We propose the following instantiation of (1),

$$\vec{w}^T \Psi(x, y, s) = \\ \frac{1}{N(x)} \sum_{j \in s} y \cdot \vec{w}_{pol}^T \psi_{pol}(x^j) + \vec{w}_{subj}^T \psi_{subj}(x^j), \tag{3}$$

where the first term in the summation captures the quality of predicting polarity y on sentences in s, the second term captures the quality of predicting s as the subjective sentences, and $N(x)$ is a normalizing factor (which will be discussed in more detail in Section 4.3). We represent the weight vector as

$$\vec{w} = \begin{bmatrix} \vec{w}_{pol} \\ \vec{w}_{subj} \end{bmatrix}, \tag{4}$$

and $\psi_{pol}(x^j)$ and $\psi_{subj}(x^j)$ denote the polarity and subjectivity features of sentence x^j, respectively. Note that ψ_{pol} and ψ_{subj} are disjoint by construction, i.e., $\psi_{pol}^T \psi_{subj} = 0$. We will present extensions in Section 4.5.

For example, suppose ψ_{pol} and ψ_{subj} were both bag-of-words feature vectors. Then we might learn a high weight for the feature corresponding to the word "think" in ψ_{subj} since that word is indicative of the sentence being subjective (but not necessarily indicating positive or negative polarity).

4.1 Making Predictions

Algorithm 1 describes our inference procedure. Recall from (2) that our hypothesis function predicts the sentiment label that maximizes (3). To do this, we compare the best set of sentences that explains a positive polarity prediction with the best set that explains a negative polarity prediction.

We now specify the structure of $S(x)$. In this paper, we use a cardinality constraint,

$$S(x) = \{s \subseteq \{1, \ldots, |x|\} \ : \ |s| \leq f(|x|)\}, \tag{5}$$

Algorithm 2 Training Algorithm for OP 1

```
1: Input: $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ //training data
2: Input: $C$ //regularization parameter
3: Input: $(s_1, \ldots, s_N)$ //initial guess
4: $\vec{w} \leftarrow$ SSVMSolve($C, \{(x_i, y_i, s_i)\}_{i=1}^{N}$)
5: while $\vec{w}$ not converged do
6:     for $i = 1, \ldots, N$ do
7:         $s_i \leftarrow \mathrm{argmax}_{s \in S(x_i)} \vec{w}^T \Psi(x_i, y_i, s)$
8:     end for
9:     $\vec{w} \leftarrow$ SSVMSolve($C, \{(x_i, y_i, s_i)\}_{i=1}^{N}$)
10: end while
11: Return $\vec{w}$
```

where $f(|x|)$ is a function that depends only on the number of sentences in x. For example, a simple function is $f(|x|) = |x| \cdot 0.3$, indicating that at most 30% of the sentences in x can be subjective.

Using this definition of $S(x)$, we can then compute the best set of subjective sentences for each possible y by computing the joint subjectivity and polarity score of each sentence x^j in isolation,

$$y \cdot \vec{w}_{pol}^T \psi_{pol}(x^j) + \vec{w}_{subj}^T \psi_{subj}(x^j),$$

and selecting the top $f(|x|)$ as s (or fewer, if there are fewer than $f(|x|)$ that have positive joint score).

4.2 Training

For training, we will use an approach based on latent variable structural SVMs (Yu and Joachims, 2009).

Optimization Problem 1.

$$\min_{\vec{w}, \xi \geq 0} \frac{1}{2} \|\vec{w}\|^2 + \frac{C}{N} \sum_{i=1}^{N} \xi_i \tag{6}$$

s.t. $\forall i$:

$$\max_{s \in S_i} \vec{w}^T \Psi(x_i, y_i, s) \geq \max_{s' \in S(x_i)} \vec{w}^T \Psi(x_i, -y_i, s') + 1 - \xi_i \tag{7}$$

OP 1 optimizes the standard SVM training objective for binary classification. Each training example has a corresponding constraint (7), which is quantified over the best possible explanation of the training polarity label. Note that we never observe the true explanation for the training labels; they are the hidden or latent variables. The hidden variables are also ignored in the objective function.

As a result, one can interpret OP 1 to be directly optimizing a trade-off between model complexity (as measured using the 2-norm) and document-level classification error in the training set. This has two main advantages over related training approaches. First, it solves the multi-level problem jointly as opposed to separately, which avoids introducing difficult to control propagation errors. Second, it does not require solving the sentence-level task perfectly, and also does not require precise sentence-level training labels. In other words, our goal is to learn to identify the informative (subjective) sentences that best explain the training labels to the extent required for good document classification performance.

OP 1 is non-convex because of the constraints (7). To solve OP 1, we use the combination of the CCCP algorithm (Yuille and Rangarajan, 2003) with cutting plane training of structural SVMs (Joachims et al., 2009), as proposed in Yu and Joachims (2009). Suppose each constraint (7) is replaced by

$$\vec{w}^T \Psi(x_i, y_i, s_i) \geq \max_{s' \in S(x_i)} \vec{w}^T \Psi(x_i, -y_i, s') + 1 - \xi_i,$$

where s_i is some fixed explanation (e.g., an initial guess of the best explanation). Then OP 1 reduces to a standard structural SVM, which can be solved efficiently (Joachims et al., 2009). Algorithm 2 describes our training procedure. Starting with an initial guess s_i for each training example, the training procedure alternates between solving an instance of the resulting structural SVM (called *SSVMSolve* in Algorithm 2) using the currently best known explanations s_i (Line 9), and making a new guess of the best explanations (Line 7). Yu and Joachims (2009) showed that this alternating procedure for training latent variable structural SVMs is an instance of the CCCP procedure (Yuille and Rangarajan, 2003), and so is guaranteed to converge to a local optimum.

For our experiments, we do not train until convergence, but instead use performance on a validation set to choose the halting iteration. Since OP 1 is non-convex, a good initialization is necessary. To generate the initial explanations, one can use an off-the-shelf sentiment classifier such as OpinionFinder[2] (Wilson et al., 2005). For some datasets, there exist documents with annotated sentences, which we

[2] `http://www.cs.pitt.edu/mpqa/opinionfinderrelease/`

can treat either as the ground truth or another (very good) initial guess of the explanatory sentences.

4.3 Feature Representation

Like any machine learning approach, we must specify a useful set of features for the ψ vectors described above. We will consider two types of features.

Bag-of-words. Perhaps the simplest approach is to define ψ using a bag-of-words feature representation, with one feature corresponding to each word in the active lexicon of the corpus. Using such a feature representation might allow us to learn which words have high polarity (e.g., "great") and which are indicative of subjective sentences (e.g., "opinion").

Sentence properties. We can incorporate many useful features to describe sentence subjectivity. For example, subjective sentences might densely populate the end of a document, or exhibit spatial coherence (so features describing previous sentences might be useful for classifying the current sentence). Such features cannot be compactly incorporated into flat models that ignore the document structure.

For our experiments, we normalize each ψ_{subj} and ψ_{pol} to have unit 2-norm.

Joint Feature Normalization. Another design decision is the choice of normalization $N(x)$ in (3). Two straightforward choices are $N(x) = f(|x|)$ and $N(x) = \sqrt{f(|x|)}$, where $f(|x|)$ is the size constraint as described in (5). In our experiments we tried both and found the square root normalization to work better in practice; therefore all the experimental results are reported using $N(x) = \sqrt{f(|x|)}$. The appendix contains an analysis that sheds light on when square root normalization can be useful.

4.4 Incorporating Proximity Information

As mentioned in Section 4.3, it is possible (and likely) for subjective sentences to exhibit spatial coherence (e.g., they might tend to group together). To exploit this structure, we will expand the feature space of ψ_{subj} to include both the words of the current and previous sentence as follows,

$$\psi_{subj}(x, j) = \begin{bmatrix} \psi_{subj}(x^j) \\ \psi_{subj}(x^{j-1}) \end{bmatrix}.$$

The corresponding weight vector can be written as

$$\vec{w}'_{subj} = \begin{bmatrix} \vec{w}_{subj} \\ \vec{w}_{prevSubj} \end{bmatrix}.$$

By adding these features, we are essentially assuming that the words of the previous sentence are predictive of the subjectivity of the current sentence.

Alternative approaches include explicitly accounting for this structure by treating subjective sentence extraction as a sequence-labeling problem, such as in McDonald et al. (2007). Such structure formulations can be naturally encoded in the joint feature map. Note that the inference procedure in Algorthm 1 is still tractable, since it reduces to comparing the best sequence of subjective/objective sentences that explains a positive sentiment versus the best sequence that explains a negative sentiment. For this study, we chose not to examine this more expressive yet more complex structure.

4.5 Extensions

Though our initial model (3) is simple and intuitive, performance can depend heavily on the quality of latent variable initialization and the quality of the feature structure design. Consider the case where the initialization contains only objective sentences that do not convey any sentiment. Then all the features initially available during training are generated from these objective sentences and are thus useless for sentiment classification. In other words, too much useful information has been suppressed for the model to make effective decisions. To hedge against learning poor models due to using a poor initialization and/or a suboptimal feature structure, we now propose extensions that incorporate information from the entire document.

We identify the following desirable properties that any such extended model should satisfy:

(A) The model should be linear.

(B) The model should be trained jointly.

(C) The component that models the entire document should influence which sentences are extracted.

The first property stems from the fact that our approach relies on linear models. The second property is desirable since joint training avoids error propagation that can be difficult to control. The third property deals with the information suppression issue.

4.5.1 Regularizing Relative to a Prior

We first consider a model that satisfies properties (A) and (C). Using the representation in (4), we propose a training procedure that regularize $\vec{w}_{pol}$ relative to a prior model. Suppose we have a weight vector $\vec{w}_0$ which indicated the a priori guess of the contribution of each corresponding feature, then we can train our model using OP 2,

Optimization Problem 2.

$$\min_{\vec{w},\xi\geq 0} \frac{1}{2}\|\vec{w}-\vec{w}_0\|^2 + \frac{C}{N}\sum_{i=1}^{N}\xi_i$$

$$\textit{s.t.}\ \forall i:$$

$$\max_{s\in S_i} \vec{w}^T\Psi(x_i,y_i,s) \geq \max_{s'\in S(x_i)} \vec{w}^T\Psi(x_i,-y_i,s') + 1 - \xi_i$$

For our experiments, we use

$$\vec{w}_0 = \begin{bmatrix} \vec{w}_{doc} \\ 0 \end{bmatrix},$$

where $\vec{w}_{doc}$ denotes a weight vector trained to classify the polarity of entire documents. Then one can interpret OP 2 as enforcing that the polarity weights $\vec{w}_{pol}$ not be too far from $\vec{w}_{doc}$. Note that $\vec{w}_0$ must be available before training. Therefore this approach does not satisfy property (B).

4.5.2 Extended Feature Space

One simple way to satisfy all three aforementioned properties is to jointly model not only polarity and subjectivity of the extracted sentences, but also polarity of the entire document. Let $\vec{w}_{doc}$ denote the weight vector used to model the polarity of entire document x (so the document polarity score is then $\vec{w}_{doc}^T\psi_{pol}(x)$). We can also incorporate this weight vector into our structured model to compute a smoothed polarity score of each sentence via $\vec{w}_{doc}^T\psi_{pol}(x^j)$. Following this intuition, we propose the following structured model,

$$\vec{w}^T\Psi(x,y,s) = \frac{y}{N(x)}\left(\sum_{j\in s}\left(\vec{w}_{pol}^T\psi_{pol}(x^j) + \vec{w}_{doc}^T\psi_{pol}(x^j)\right)\right) + \frac{1}{N(x)}\left(\sum_{j\in s}\vec{w}_{subj}^T\psi_{subj}(x^j)\right) + y\cdot\vec{w}_{doc}^T\psi_{pol}(x)$$

where the weight vector is now

$$\vec{w} = \begin{bmatrix} \vec{w}_{pol} \\ \vec{w}_{subj} \\ \vec{w}_{doc} \end{bmatrix}.$$

Training this model via OP 1 achieves that $\vec{w}_{doc}$ is (1) used to model the polarity of the entire document, and (2) used to compute a smoothed estimate of the polarity of the extracted sentences. This satisfies all three properties (A), (B), and (C), although other approaches are also possible.

5 Experiments

5.1 Experimental Setup

We evaluate our methods using the Movie Reviews and U.S. Congressional Floor Debates datasets, following the setup used in previous work for comparison purposes.[3]

Movie Reviews. We use the movie reviews dataset from Zaidan et al. (2007) that was originally released by Pang and Lee (2004). This version contains annotated rationales for each review, which we use to generate an additional initialization during training (described below). We follow exactly the experimental setup used in Zaidan et al. (2007).[4]

U.S. Congressional Floor Debates. We also use the U.S. Congressional floor debates transcripts from Thomas et al. (2006). The data was extracted from GovTrack (http://govtrack.us), which has all available transcripts of U.S. floor debates in the House of Representatives in 2005. As in previous work, only debates with discussions of "controversial" bills were considered (where the losing side had at least 20% of the speeches). The goal is to predict the vote ("yea" or "nay") for the speaker of each speech segment. For our experiments, we evaluate our methods using the speaker-based speech-segment classification setting as described in Thomas et al. (2006).[5]

[3] Datasets in the required format for *SVM*^{sle} are available at `http://www.cs.cornell.edu/~ainur/data.html`

[4] Since the rationale annotations are available for nine out of 10 folds, we used the 10-th fold as the blind test set. We trained nine different models on subsets of size eight, used the remaining fold as the validation set, and then measured the average performance on the final test set.

[5] In the other setting described in Thomas et al. (2006) (segment-based speech-segment classification), around 39% of

Table 1: Summary of the experimental results for the Movie Reviews (top) and U.S. Congressional Floor Debates (bottom) datasets using *SVM*^{sle}, *SVM*^{sle} w/ Prior and *SVM*$^{sle}_{fs}$ with and without proximity features.

INITIALIZATION	*SVM*^{sle}	+ Prox.Feat.	*SVM*^{sle} w/ Prior	+ Prox.Feat.	*SVM*$^{sle}_{fs}$	+ Prox.Feat.
Random 30%	87.22	85.44	87.61	87.56	89.50	88.22
Last 30%	89.72 $*$	88.83	90.50 $*$	90.00 $*$	91.06 $\star$	91.22 $\star$
OpinionFinder	91.28 $\star$	90.89 $\star$	91.72 $\star$	93.22 $\star$	92.50 $\star$	92.39 $\star$
Annot.Rationales	91.61 $\star$	92.00 $\star$	92.67 $\star$	92.00 $\star$	92.28 $\star$	93.22 $\star$

INITIALIZATION	*SVM*^{sle}	+ Prox.Feat.	*SVM*^{sle} w/ Prior	+ Prox.Feat.	*SVM*$^{sle}_{fs}$	+ Prox.Feat.
Random 30%	78.84	73.14	78.49	76.40	77.33	73.84
Last 30%	73.26	73.95	71.51	73.60	67.79	73.37
OpinionFinder	77.33	79.53	77.09	78.60	77.67	77.09

– For Movie Reviews, the SVM baseline accuracy is 88.56%. A $\star$ (or $*$) indicates statically significantly better performance than baseline according to the paired t-test with $p < 0.001$ (or $p < 0.05$).
– For U.S. Congressional Floor Debates, the SVM baseline accuracy is 70.00%. Statistical significance cannot be calculated because the data comes in a single split.

Since our training procedure solves a non-convex optimization problem, it requires an initial guess of the explanatory sentences. We use an explanatory set size (5) of 30% of the number of sentences in each document, $L = \lceil 0.3 \cdot |x| \rceil$, with a lower cap of 1. We generate initializations using OpinionFinder (Wilson et al., 2005), which were shown to be a reasonable substitute for human annotations in the Movie Reviews dataset (Yessenalina et al., 2010).[6]

We consider two additional (baseline) methods for initialization: using a random set of sentences, and using the last 30% of sentence in the document. In the Movie Reviews dataset, we also use sentences containing human-annotator rationales as a final initialization option. No such manual annotations are available for the Congressional Debates.

5.2 Experimental Results

We evaluate three versions of our model: the initial model (3) which we call *SVM*^{sle} (**SVM**s for **S**entiment classification with **L**atent **E**xplanations), *SVM*^{sle} regularized relative to a prior as described in Section 4.5.1 which we refer to as *SVM*^{sle} w/ Prior,[7] and the feature smoothing model described in Section 4.5.2 which we call *SVM*$^{sle}_{fs}$. Due to the difficulty of selecting a good prior, we expect *SVM*$^{sle}_{fs}$ to exhibit the most robust performance.

Table 1 shows a comparison of our proposed methods on the two datasets. We observe that *SVM*$^{sle}_{fs}$ provides both strong and robust performance. The performance of *SVM*^{sle} is generally better when trained using a prior than not in the Movie Reviews dataset. Both extensions appear to hurt performance in the U.S. Congressional Floor Debates dataset. Using OpinionFinder to initialize our training procedure offers good performance across both datasets, whereas the baseline initializations exhibit more erratic performance behavior.[8] Unsurprisingly, initializing using human annotations (in the Movie Reviews dataset) can offer further improvement. Adding proximity features (as described in Section 4.4) in general seems to improve performance when using a good initialization, and hurts performance otherwise.

the documents in the whole dataset contain only 1-3 sentences, making it an uninteresting setting to analyze with our model.

[6] We select all sentences whose majority vote of OpinionFinder word-level polarities matches the document's sentiment. If there are fewer than L sentences, we add sentences starting from the end of the document. If there are more, we remove sentences starting from the beginning of the document.

[7] We either used the same value of C to train both standard SVM model and *SVM*^{sle} w/ Prior or used the best standard SVM model on the validation set to train *SVM*^{sle} w/ Prior. We chose the combination that works the best on the validation set.

[8] Using the random initialization on the U.S. Congressional Floor Debates dataset offers surprisingly good performance.

Table 2: Comparison of SVM^{sle}_{fs} with previous work on the Movie Reviews dataset. We considered two settings: when human annotations are available (Annot. Labels), and when they are unavailable (No Annot. Labels).

	METHOD	ACC
Baseline	SVM	88.56
Annot. Labels	Zaidan et al. (2007)	92.20
	SVM^{sle}_{fs}	92.28
	SVM^{sle}_{fs}+ Prox.Feat.	93.22
No Annot. Labels	Yessenalina et al. (2010)	91.78
	SVM^{sle}_{fs}	92.50
	SVM^{sle}_{fs}+Prox.Feat.	92.39

Table 3: Comparison of SVM^{sle}_{fs} with previous work on the U.S. Congressional Floor Debates dataset for the speaker-based segment classification task.

	METHOD	ACC
Baseline	SVM	70.00
Prior work	Thomas et al. (2006)	71.28
	Bansal et al. (2008)	75.00
Our work	SVM^{sle}_{fs}	77.67
	SVM^{sle}_{fs}+ Prox.Feat.	77.09

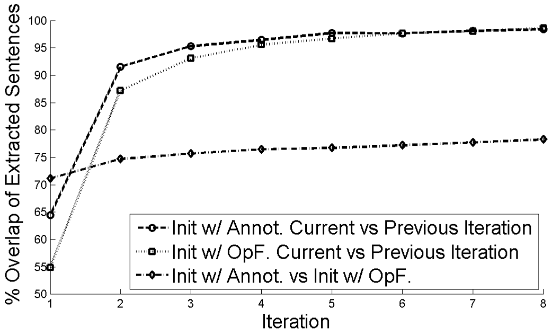


Figure 1: Overlap of extracted sentences from different SVM^{sle}_{fs} models on the Movie Reviews training set.

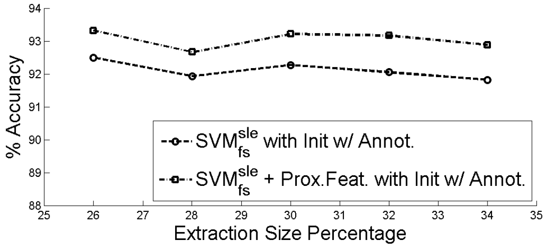


Figure 2: Test accuracy on the Movie Reviews dataset for SVM^{sle}_{fs} while varying extraction size.

Tables 2 and 3 show a comparison of SVM^{sle}_{fs} with previous work on the Movie Reviews and U.S. Congressional Floor Debates datasets, respectively. For the Movie Reviews dataset, we considered two settings: when human annotations are available, and when they are not (in which case we initialized using OpinionFinder). For the U.S. Congressional Floor Debates dataset we used only the latter setting, since there are no annotations available for this dataset. In all cases we observe SVM^{sle}_{fs} showing improved performance compared to previous results.

Training details. We tried around 10 different values for C parameter, and selected the final model based on the validation set. The training procedure alternates between training a standard structural SVM model and using the subsequent model to re-label the latent variables. We selected the halting iteration of the training procedure using the validation set. When initializing using human annotations for the Movie Reviews dataset, the halting iteration is typically the first iteration, whereas the halting iteration is typically chosen from a later iteration when initializing using OpinionFinder.

Figure 1 shows the per-iteration overlap of extracted sentences from SVM^{sle}_{fs} models initialized using OpinionFinder and human annotations on the Movie Reviews training set. We can see that training has approximately converged after about 10 iterations.[9] We can also see that both models iteratively learn to extract sentences that are more similar to each other than their respective initializations (the overlap between the two initializations is 57%). This is an indicator that our learning problem, despite being non-convex and having multiple local optima, has a reasonably large “good” region that can be approached using different initialization methods.

Varying the extraction size. Figure 2 shows how accuracy on the test set of SVM^{sle}_{fs} changes on the Movie Reviews dataset as a function of varying the extraction size $f(|x|)$ from (5). We can see that performance changes smoothly[10] (and so is robust), and that one might see further improvement from more

[9] The number of iterations required to converge is an upper bound on the number of iterations from which to choose the halting iteration (based on a validation set).

[10] The smoothness will depend on the initialization.

Table 4: Example ”yea” speech with ***Latent Explanations*** from the U.S. Congressional Floor Debates dataset predicted by SVM^{sle}_{fs} with OpinionFinder initialization. Latent Explanations are preceded by solid circles with numbers denoting their preference order (1 being most preferred by SVM^{sle}_{fs}). The five least subjective sentences are preceded by circles with numbers denoting the subjectivity order (1 being least subjective according to SVM^{sle}_{fs}).

❷ Mr. Speaker, I am proud to stand on the house floor today to speak in favor of the Stem Cell Research Enhancement Act, legislation which will bring hope to millions of people suffering from disease in this nation. ❸ I want to thank Congresswoman Degette and Congressman Castle for their tireless work in bringing this bill to the house floor for a vote.

① The discovery of embryonic stem cells is a major scientific breakthrough. ⑤ Embryonic stem cells have the potential to form any cell type in the human body. This could have profound implications for diseases such as Alzheimer’s, Parkinson’s, various forms of brain and spinal cord disorders, diabetes, and many types of cancer. ② According to the Coalition for the Advancement of Medical Research, there are at least 58 diseases which could potentially be cured through stem cell research.

That is why more than 200 major patient groups, scientists, and medical research groups and 80 Nobel Laureates support the Stem Cell Research Enhancement Act. ③ They know that this legislation will give us a chance to find cures to diseases affecting 100 million Americans.

I want to make clear that I oppose reproductive cloning, as we all do. I have voted against it in the past. ***❹ However, that is vastly different from stem cell research and as an ovarian cancer survivor, I am not going to stand in the way of science.***

Permitting peer-reviewed Federal funds to be used for this research, combined with public oversight of these activities, is our best assurance that research will be of the highest quality and performed with the greatest dignity and moral responsibility. The policy President Bush announced in August 2001 has limited access to stem cell lines and has stalled scientific progress.

As a cancer survivor, I know the desperation these families feel as they wait for a cure. ④ This congress must not stand in the way of that progress. ***❺ We have an opportunity to change the lives of millions, and I hope we take it. ❶ I urge my colleagues to support this legislation.***

careful tuning of the size constraint.

Examining an example prediction. Our proposed methods are not designed to extract interpretable explanations, but examining the extracted explanations might still yield meaningful information. Table 4 contains an example speech from the U.S. Congressional Floor Debates test set, with Latent Explanations found by SVM^{sle}_{fs} highlighted in boldface. This speech was made in support of the Stem Cell Research Enhancement Act. For comparison, Table 4 also shows the five least subjective sentences according to SVM^{sle}_{fs}. Notice that most of these “objective” sentences can plausibly belong to speeches made in opposition to bills that limit stem cell research funding. That is, they do not clearly indicate the speaker’s stance towards the specific bill in question. We can thus see that our approach can indeed learn to infer sentences that are essential to understanding the document-level sentiment.

6 Discussion

Making good structural assumptions simplifies the development process. Compared to methods that modify the training of flat document classifiers (e.g., Zaidan et al. (2007)), our approach uses fewer parameters, leading to a more compact and faster training stage. Compared to methods that use a cascaded approach (e.g., Pang and Lee (2004)), our approach is more robust to errors in the lower-level subtask due to being a joint model.

Introducing latent variables makes the training procedure more flexible by not requiring lower-level labels, but does require a good initialization (i.e., a reasonable substitute for the lower-level labels). We believe that the widespread availability of off-the-shelf sentiment lexicons and software, despite being developed for a different domain, makes this issue less of a concern, and in fact creates an opportunity for approaches like ours to have real impact.

One can incorporate many types of sentence-level information that cannot be directly incorporated into a flat model. Examples include scores from another sentence-level classifier (e.g., from Nakagawa et. al (2010)) or combining phrase-level polarity scores (e.g., from Choi and Cardie (2008)) for each sentence, or features that describe the position of the sentence in the document.

Most prior work on the U.S. Congressional Floor Debates dataset focused on using relationships between speakers such as agreement (Thomas et al., 2006; Bansal et al., 2008), and used a global mincut inference procedure. However, they require all test instances to be known in advance (i.e., their formulations are transductive). Our method is not limited to the transductive setting, and instead exploits a different and complementary structure: the latent explanation (i.e., only some sentences in the speech are indicative of the speaker's vote).

In a sense, the joint feature structure used in our model is the simplest that could be used. Our model makes no explicit structural dependencies between sentences, so the choice of whether to extract each sentence is essentially made independently of other sentences in the document. More sophisticated structures can be used if appropriate. For instance, one can formulate the sentence extraction task as a sequence labeling problem similar to (McDonald et al., 2007), or use a more expressive graphical model such as in (Pang and Lee, 2004; Thomas et al., 2006). So long as the global inference procedure is tractable or has a good approximation algorithm, then the training procedure is guaranteed to converge with rigorous generalization guarantees (Finley and Joachims, 2008). Since any formulation of the extraction subtask will suppress information for the main document-level task, one must take care to properly incorporate smoothing if necessary.

Another interesting direction is training models to predict not only sentiment polarity, but also whether a document is objective. For example, one can pose a three class problem ("positive", "negative", "objective"), where objective documents might not necessarily have a good set of (subjective) explanatory sentences, similar to (Chang et al., 2010).

7 Conclusion

We have presented latent variable structured models for the document sentiment classification task. These models do not rely on sentence-level annotations, and are trained jointly (over both the document and sentence levels) to directly optimize document-level accuracy. Experiments on two standard sentiment analysis datasets showed improved performance over previous results.

Our approach can, in principle, be applied to any classification task that is well modeled by jointly solving an extraction subtask. However, as evidenced by our experiments, proper training does require a reasonable initial guess of the extracted explanations, as well as ways to mitigate the risk of the extraction subtask suppressing too much information (such as via feature smoothing).

Acknowledgments

This work was supported in part by National Science Foundation Grants BCS-0904822, BCS-0624277, IIS-0535099; by a gift from Google; and by the Department of Homeland Security under ONR Grant N0014-07-1-0152. The second author was also supported in part by a Microsoft Research Graduate Fellowship. The authors thank Yejin Choi, Thorsten Joachims, Nikos Karampatziakis, Lillian Lee, Chun-Nam Yu, and the anonymous reviewers for their helpful comments.

Appendix

Recall that all the ψ_{subj} and ψ_{pol} vectors have unit 2-norm, which is assumed here to be desirable. We now show that using $N(x) = \sqrt{f(|x|)}$ achieves a similar property for $\Psi(x, y, s)$. We can write the squared 2-norm of $\Psi(x, y, s)$ as

$$|\Psi(x,y,s)|^2 = \frac{1}{N(x)^2}\left[\sum_{j\in s} y\cdot\psi_{pol}(x^j) + \psi_{subj}(x^j)\right]^2$$
$$= \frac{1}{f(|x|)}\left[\left(\sum_{j\in s}\psi_{pol}(x^j)\right)^2 + \left(\sum_{j\in s}\psi_{subj}(x^j)\right)^2\right],$$

where the last equality follows from the fact that

$$\psi_{pol}(x^j)^T\psi_{subj}(x^j) = 0,$$

due to the two vectors using disjoint feature spaces by construction. The summation of the $\psi_{pol}(x^j)$ terms is written as

$$\begin{aligned}\left(\sum_{j\in s}\psi_{pol}(x^j)\right)^2 &= \sum_{j\in s}\sum_{i\in s}\psi_{pol}(x^j)^T\psi_{pol}(x^i)\\ &\approx \sum_{j\in s}\psi_{pol}(x^j)^T\psi_{pol}(x^j) \qquad (8)\\ &= \sum_{j\in s} 1 \le f(|x|),\end{aligned}$$

where (8) follows from the sparsity assumption that

$$\forall i \neq j:\ \psi_{pol}(x^j)^T\psi_{pol}(x^i) \approx 0.$$

A similar argument applies for the $\psi_{subj}(x^j)$ terms. Thus, by choosing $N(x) = \sqrt{f(|x|)}$ the joint feature vectors $\Psi(x, y, s)$ will have approximately equal magnitude as measured using the 2-norm.

References

Mohit Bansal, Claire Cardie, and Lillian Lee. 2008. The power of negative thinking: Exploiting label disagreement in the min-cut classification framework. In *International Conference on Computational Linguistics (COLING)*.

Ming-Wei Chang, Dan Goldwasser, Dan Roth, and Vivek Srikumar. 2010. Discriminative learning over constrained latent representations. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Yejin Choi and Claire Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Empirical Methods in Natural Language Processing (EMNLP)*.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *ACL Conference on Natural Language Learning (CoNLL)*, July.

Pedro Felzenszwalb, David McAllester, and Deva Ramanan. 2008. A discriminatively trained, multiscale, deformable part model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Thomas Finley and Thorsten Joachims. 2008. Training structural svms when exact inference is intractable. In *International Conference on Machine Learning (ICML)*.

Thorsten Joachims, Thomas Finley, and Chun-Nam Yu. 2009. Cutting plane training of structural svms. *Machine Learning*, 77(1):27–59.

Yi Mao and Guy Lebanon. 2006. Isotonic conditional random fields and local sentiment flow. In *Neural Information Processing Systems (NIPS)*.

Ryan McDonald, Kerry Hannan, Tyler Neylon, Mike Wells, and Jeff Reynar. 2007. Structured models for fine-to-coarse sentiment analysis. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using crfs with hidden variables. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Slav Petrov and Dan Klein. 2007. Discriminative log-linear grammars with latent variables. In *Neural Information Processing Systems (NIPS)*.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from Congressional floor-debate transcripts. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Ainur Yessenalina, Yejin Choi, and Claire Cardie. 2010. Automatically generating annotator rationales to improve sentiment classification. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Chun-Nam Yu and Thorsten Joachims. 2009. Learning structural svms with latent variables. In *International Conference on Machine Learning (ICML)*.

Alan L. Yuille and Anand Rangarajan. 2003. The concave-convex procedure. *Neural Computation*, 15(4):915–936, April.

Omar F. Zaidan and Jason Eisner. 2008. Modeling annotators: a generative approach to learning from annotator rationales. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Omar F. Zaidan, Jason Eisner, and Christine Piatko. 2007. Using "annotator rationales" to improve machine learning for text categorization. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Cross Language Text Classification by Model Translation and Semi-Supervised Learning

Lei Shi
Yahoo! Global R&D
Beijing, China
lshi@yahoo-inc.com

Rada Mihalcea
University of North Texas
Denton, TX, U.S.A.
rada@cs.unt.edu

Mingjun Tian
Yahoo! Global R&D
Beijing, China
mingjun@yahoo-inc.com

Abstract

In this paper, we introduce a method that automatically builds text classifiers in a new language by training on already labeled data in another language. Our method transfers the classification knowledge across languages by translating the model features and by using an Expectation Maximization (EM) algorithm that naturally takes into account the ambiguity associated with the translation of a word. We further exploit the readily available unlabeled data in the target language via semi-supervised learning, and adapt the translated model to better fit the data distribution of the target language.

1 Introduction

Given the accelerated growth of the number of multilingual documents on the Web and elsewhere, the need for effective multilingual and cross-lingual text processing techniques is becoming increasingly important. There is a growing number of methods that use data available in one language to build text processing tools for another language, for diverse tasks such as word sense disambiguation (Ng et al., 2003), syntactic parsing (Hwa et al., 2005), information retrieval (Monz and Dorr, 2005), subjectivity analysis (Mihalcea et al., 2007), and others.

In this paper, we address the task of *cross-lingual text classification* (CLTC), which builds text classifiers for multiple languages by using training data in one language, thereby avoiding the costly and time-consuming process of labeling training data for each individual language. The main idea underlying our approach to CLTC is that although content can be expressed in different forms in different languages, there is a significant amount of knowledge that is shared for similar topics that can be effectively used to port topic classifiers across languages.

Previous methods for CLTC relied mainly on machine translation, by translating the training data into the language of the test data or vice versa, so that both training and test data belong to the same language. Monolingual text classification algorithms can then be applied on these translated data. Although intuitive, these methods suffer from two major drawbacks.

First, most off-the-shelf machine translation systems typically generate only their best translation for a given text. Since machine translation is known to be a notoriously hard problem, applying monolingual text classification algorithms directly on the erroneous translation of training or test data may severely deteriorate the classification accuracy.

Second, similar to domain adaptation in statistical machine learning, due to the discrepancy of data distribution between the training domain and test domain, data distribution across languages may vary because of the difference of culture, people's interests, linguistic expression in different language regions. So even if the translation of training or test data is perfectly correct, the cross language classifier may not perform as well as the monolingual one trained and tested on the data from the same language.

In this paper, we propose a new approach to CLTC, which trains a classification model in the source language and ports the model to the target language, with the translation knowledge learned using the EM algorithm. Unlike previous methods based on machine translation (Fortuna and Shawe-Taylor, 2005), our method takes into account dif-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1057–1067,
MIT, Massachusetts, USA, 9-11 October 2010.

ferent possible translations for model features. The translated model serves as an initial classifier for a semi-supervised process, by which the model is further adjusted to fit the distribution of the target language. Our method does not require any labeled data in the target language, nor a machine translation system. Instead, the only requirement is a reasonable amount of unlabeled data in the target language, which is often easy to obtain.

In the following sections, we first review related work. In section 3, we introduce our method that translates the classification model with the translation knowledge learned using the EM algorithm. Section 4 describes model adaptation by training the translated model with unlabeled documents in the target language. Experiments and evaluations are presented in section 5 and finally we conclude the paper in section 6.

2 Related Work

Text classification has rightfully received a lot of attention from both the academic and industry communities, being one of the areas in natural language processing that has a very large number of practical applications. Text classification techniques have been applied to many diverse problems, ranging from topic classification (Joachims, 1997), to genre detection (Argamon et al., 1998), opinion identification (Pang and Lee, 2004), spam detection (Sahami et al., 1998), gender and age classification (Schler et al., 2006).

Text classification is typically formulated as a learning task, where a classifier learns how to distinguish between categories in a given set, using features automatically extracted from a collection of documents. In addition to the learning methodology itself, the accuracy of the text classifier also depends to a large extent upon the amount of training data available at hand. For instance, distinguishing between two categories for which thousands of manually annotated examples are already available is expected to perform better than trying to separate categories that have only a handful of labeled documents.

Some of the most successful approaches to date for text classification involve the use of machine learning methods, which assume that enough annotated data is available such that a classification model can be automatically learned. These include algorithms such as Naive Bayes (Joachims, 1997; McCallum and Nigam, 1998), Rocchio classifiers (Joachims, 1997; Moschitti, 2003), Maximum Entropy (Nigam et al., 1999) or Support Vector Machines (Vapnik, 1995; Joachims, 1998). If only a small amount of annotated data is available, the alternative is to use semi-supervised bootstrapping methods such as co-training or self-training, which can also integrate raw unlabeled data into the learning model (Blum and Mitchell, 1998; Nigam and Ghani, 2000).

Despite the attention that monolingual text classification has received from the research community, there is only very little work that was done on cross-lingual text classification. The work that is most closely related to ours is (Gliozzo and Strapparava, 2006), where a multilingual domain kernel is learned from comparable corpora, and subsequently used for the cross-lingual classification of texts. In experiments run on Italian and English, Gliozzo and Strapparava showed that the multilingual domain kernel exceeds by a large margin a bag-of-words approach. Moreover, they demonstrated that the use of a bilingual dictionary can drastically improve the performance of the models learned from corpora.

(Fortuna and Shawe-Taylor, 2005; Olsson et al., 2005) studied the use of machine translation tools for the purpose of cross language text classification and mining. These approaches typically translate the training data or test data into the same language, followed by the application of a monolingual classifier. The performance of such classifiers very much depends on the quality of the machine translation tools. Unfortunately, the development of statistical machine translation systems (Brown et al., 1993) is hindered by the lack of availability of parallel corpora and the quality of their output is often erroneous. Several methods were proposed (Shi et al., 2006; Nie et al., 1999) to automatically acquire a large quantity of parallel sentences from the web, but such web data is however predominantly confined to a limited number of domains and language pairs.

(Dai et al., 2007) experimented with the use of transfer learning for text classification. Although in this method the transfer learning is performed across different domains in the same language, the underlying principle is similar to CLTC in the sense that different domains or languages may share a significant amount of knowledge in similar classification tasks. (Blum and Mitchell, 1998) employed semi-supervised learning for training text classifiers. This method bootstraps text classifiers with only unlabeled data or a small amount of labeled training data, which is close to our setting that tries to leverage labeled data and unlabeled data in different languages to build text classifiers.

Finally, also closely related is the work carried out in the field of sentiment and subjectivity analysis for cross-lingual classification of opinions. For instance, (Mihalcea et al., 2007) use an English corpus annotated for subjectivity along with parallel text to build a subjectivity classifier for Romanian. Similarly, (Banea et al., 2008) propose a method based on machine translation to generate parallel texts, followed by a cross-lingual projection of subjectivity labels, which are used to train subjectivity annotation tools for Romanian and Spanish. A related, yet more sophisticated technique is proposed in (Wan, 2009), where a co-training approach is used to leverage resources from both a source and a target language. The technique is tested on the automatic sentiment classification of product reviews in Chinese, and showed to successfully make use of both cross-language and within-language knowledge.

3 Cross Language Model Translation

To make the classifier applicable to documents in a foreign language, we introduce a method where model features that are learned from the training data are translated from the source language into the target language. Using this translation process, a feature associated with a word in the source language is transferred to a word in the target language so that the feature is triggered when the word occurs in the target language test document.

In a typical translation process, the features would be translated by making use of a bilingual dictionary. However, this translation method has a major drawback, due to the ambiguity usually associated with the entries in a bilingual dictionary: a word in one language can have multiple translations in another language, with possibly disparate meanings. If an incorrect translation is selected, it can distort the classification accuracy, by introducing erroneous features into the learning model. Therefore, our goal is to minimize the distortion during the model translation process, in order to maximize the classification accuracy in the target language.

In this paper, we introduce a method that employs the EM algorithm to automatically learn feature translation probabilities from labeled text in the source language and unlabeled text in the target language. Using the feature translation probabilities, we can derive a classification model for the target language from a mixture model with feature translations.

3.1 Learning Feature Translation Probabilities with EM Algorithm

Given a document d from the document collection D in the target language, the probability of generating the document $P(d)$ is the mixture of generating d with different classes $c \in C$:

$$P(d) = \sum_c P(d|c)P(c)$$

In our cross-lingual setting, we view the generation of d given a class c as a two step process. In the first step, a pseudo-document d' is generated in the source language, followed by a second step, where d' is translated into the observed document d in the target language. In this generative model, d' is a latent variable that cannot be directly observed. Since d could have multiple translations d' in the source language, the probability of generating d can then be reformulated as a mixture of probabilities as in the following equation.

$$P(d) = \sum_c P(c) \sum_{d'} P(d|d', c)P(d'|c)$$

According to the bag-of-words assumption, the document translation probability $P(d|d', c)$ is the product of the word translation probabilities $P(w_i|w'_i, c)$, where w'_i in d' is the source language word that w_i is translated from. $P(d'|c)$ is the product of $P(w'_i|c)$. The formula is rewritten as:

$$P(d) = \sum_c P(c) \sum_{d'} \prod_{i=1}^{l} P(w_i|w'_i, c)P(w'_i|c)$$

where w_i is the i^{th} word of the document d with l words. The prior probability $P(c)$ and the probability of the source language word w' given class c are estimated using the labeled training data in the source language, so we use them as known parameters. $P(w_i|w'_i, c)$ is the probability of translating the word w'_i in the source language to the word w_i in the target language given class c, and these are the parameters we want to learn from the corpus in the target language.

Using the Maximum Likelihood Estimation (MLE) framework, we learn the model parameters θ – the translation probability $P(w_i|w'_i, c)$ – by maximizing the log likelihood of a collection of documents in the target language:

$$\begin{aligned} \widehat{\theta} &= argmax_\theta \sum_{j=1}^{m} log(P(d_j, \theta)) \\ &= argmax_\theta \sum_{j=1}^{m} log(\sum_{c} P(c) \sum_{d'} \\ &\quad \prod_{i=1}^{l_j} P(w_i|w'_i, c) P(w'_i|c)) \end{aligned}$$

where m is the number of documents in the corpus in the target language and l_j is the number of words in the document d_j.

In order to estimate the optimal values of the parameters, we use the EM algorithm (Dempster et al., 1977). At each iteration of EM we determine those values by maximizing the expectation using the parameters from the previous iteration and this iterative process stops when the change in the parameters is smaller than a given threshold. We can repeat the following two steps for the purpose above.

- E-step

$$\begin{aligned} P(w'c|w) &\leftarrow \frac{P(cw'w)}{P(w)} \\ &= \frac{P(w|w'c)P(w'c)}{\sum_c \sum_{w'} P(w|w'c)P(w'c)} \end{aligned} \tag{1}$$

- M-step

$$P(w|w'c) \leftarrow \frac{f(w)P(w'c|w)}{\sum_{w \in K} f(w)P(w'c|w)} \tag{2}$$

Here $f(w)$ is the occurrence frequency of the word w in the corpus. K is the set of translation candidates in the target language for the source language word w' according to the bilingual lexicon. P(w'c) is the probability of occurrence of the source language word w' under the class c. It can be estimated from the labeled source language training data available as follows and it is regarded as a known parameter of the model.

$$P(w'c) = \frac{f(w'c)}{\sum_{w' \in V} f(w'c)} \tag{3}$$

where V is the vocabulary of the source language. Algorithm 1 illustrates the EM learning process, where $n_{w'}$ denotes the number of translation candidates for w' according to the bilingual lexicon.

Our method requires no labeled training data in the target language. Many statistical machine translation systems such as IBM models (Brown et al., 1993) learn word translation probabilities from millions of parallel sentences which are mutual translations. However, large scale parallel corpora rarely exist for most language pairs. (Koehn and Knight, 2000) proposed to use the EM algorithm to learn word translation probabilities from non-parallel monolingual corpora. However, this method estimates only class independent translation probabilities $P(w_i|w'_i)$, while our approach is able to learn class specific translation probabilities

Algorithm 1 EM algorithm for learning translation probabilities

```
$D_l \leftarrow$ labeled data in the source language
$D_u \leftarrow$ unlabeled data in the target language
$L \leftarrow$ bilingual lexicon
Initialize $P_0(w|w'c) = \frac{1}{n_{w'}}$, where $(w, w') \in L$, otherwise $P_0(w|w'c) = 0$;
Compute $P(w'c)$ with $D_l$ according to equation 3
repeat
    Calculate $P_t(w'c|w)$ with $D_u$ based on $P_{t-1}(w|w'c)$ according to equation 1
    Calculate $P_t(w|w'c)$ based on $P_{t-1}(w'c|w)$ according to equation 2
until change of $P(w|w'c)$ is smaller than the threshold
return $P(w|w'c)$
```

$P(w_i|w'_i, c)$ by leveraging available labeled training data in the source language. For example, the probability of translating “bush” as “树丛” (small trees) is higher than translating as “布什” (U.S. president) when the category of the text is “botany.”

3.2 Model Translation

In order to classify documents in the target language, a straightforward approach to transferring the classification model learned from the labeled source language training data is to translate each feature from the bag-of-words model according to the bilingual lexicon. However, because of the translation ambiguity of each word, a model in the source language could be potentially translated into many different models in the target language. Thus, we think of the probability of the class of a target language document as the mixture of the probabilities by each translated model from the source language model, weighed by their translation probabilities.

$$P(c|d, m_t) \approx \textstyle\sum_{m'_t} P(m'_t|m_s, c)P(c|d, m'_t)$$

where m_t is the target language classification model and m'_t is a candidate model translated from the model m_s trained on the labeled training data in the source language. This is a very generic representation for model translation and the model m could be any type of text classification. Specifically in this paper, we take the Maximum Entropy (ME) model(Berger et al., 1996) as an example for the model translation across languages, since the ME model is one of the most widely used text classification models. The maximum entropy classifier takes the form

$$P(c|d) = \frac{1}{Z(d)} \prod_{w \in V} e^{\lambda_w f(w,c)}$$

where: V is the vocabulary of the language; $f(w, c)$ is the feature function associated with the word w and class c and its value is set to 1 when w occurs in d and the class is c or otherwise 0. λ_w is the feature weight for $f(w_i, c)$ indicating the importance of the feature in the model. During model translation, the feature weight for $f(w_i, c)$ is transferred to $f(w'_i, c)$ in the target language model, where w'_i is the translation of w_i. $Z(d)$ is the normalization factor which is invariant to c and hence we can omit it for classification since our objective is to find the best c. According to the formulation of the Maximum Entropy model, the document can be classified as follows.

$$\widehat{c} \;=\; argmax_{c \in C} \sum_{m'_t} P(m'_t|m_s, c) \prod_{i=1}^{v} e^{\lambda_{w_s^i} f(w_t^i, c)}$$

The model translation probability $P(m'_t|m_s, c)$ can be modeled as the product of the translation probabilities of each of its individual bag-of-words features $P(m'_t|m_s, c) \approx \prod_{i=1}^{l} P(w_t^i|w_s^i, c)$ and the classification model can be further written as

$$\widehat{c} \;=\; argmax_{c \in C} \sum_{m'_t} \prod_{i=1}^{v} P(w_t^i|w_s^i, c) e^{\lambda_{w_s^i} f(w_t^i, c)}$$

where feature translation probabilities $P(w_t^i|w_s^i, c)$ are estimated with the EM algorithm described in the previous section. Note that if the average number of translations for a word w is n and v is the number of words in the vocabulary there are n^v possible models m'_t translated from m_s. However, we can do the following mathematical transformation on the equation which leads to a polynomial time complexity algorithm. The idea is that instead of enumerating the exponential number of different translations of the entire model, we will instead handle one feature at a time.

$$\sum_{m'_t} \prod_{i=1}^{v} P(w_t^i|w_s^i, c) e^{\lambda_{w_s^i} f(w_t^i, c)} =$$
$$\sum_{j=1}^{n_1} P(w_t^{1j}|w_s^1, c) e^{\lambda_1 f(w_t^{1j}, c)} \sum_{m_t^{2,v}} \prod_{i=2}^{v} P(w_t^i|w_s^i, c) e^{\lambda_i f(w_t^i, c)}$$

Here w_1 is the first word in the vocabulary of the source language and w_{1j} is a translation of w_1 in the target language with n denoting the number its translations according to the bilingual lexicon. $\sum_{m_t^{2,v}}$ are all the target language models translated from the model consisting of the rest of the words w_2 ... w_v in the source language. This process is recursive until the last word w_s^v of the vocabulary and this transforms the equation into a polynomial form as follows.

$$\begin{aligned}&\sum_{m'_t}\prod_{i=1}^{v} P(w_t^i|w_s^i,c)e^{\lambda_{w_s^i} f(w_t^i,c)}\\ &= \prod_{i=1}^{v}\sum_{j=1}^{n_i} P(w_t^{ij}|w_s^i,c)e^{\lambda_{w_s^i} f(w_t^{ij},c)}\end{aligned}$$

Based on the above transformation, the class $\widehat{c}$ for the target language document d is then calculated with the following equation.

$$\widehat{c} = argmax_{c\in C}\prod_{i=1}^{v}\sum_{j=1}^{n_i} P(w_t^{ij}|w_s^i,c)e^{\lambda_{w_s^i} f(w_t^{ij},c)}$$

The time complexity of computing the above equation is $n \times v$.

4 Model Adaptation with Semi-Supervised Learning

In addition to translation ambiguity, another challenge in building a classifier using training data in a foreign language is the discrepancy of data distribution in different languages. Direct application of a classifier translated from a foreign model may not fit well the distribution of the current language. For example, a text about "sports" in (American) English may talk about "American football," "baseball," and "basketball," whereas Chinese tend to discuss about "soccer" or "table tennis."

To alleviate this problem, we employ semi-supervised learning in order to adapt the model to the target language. Specifically, we first start by using the translated classifier from English as an initial classifier to label a set of Chinese documents. The initial classifier is able to correctly classify a number of unlabeled Chinese documents with the knowledge transferred from English training data. For instance, words like "game(比赛)," "score(比分)," "athlete(运动员)," learned from English can still effectively classify Chinese documents. We then pick a set of labeled Chinese documents with high confidence to train a new Chinese classifier. The new classifier can then learn new knowledge from these Chinese documents. E.g. it can discover that words like "soccer(足球)" or "badminton(羽毛球)" occur frequently in the Chinese "sports" documents, while words that are frequently occurring in English documents such as "superbowl(超级碗)" and "NHL(全美冰球联盟)" do not occur as often. Re-training the classifier with the Chinese documents can adjust the feature weights for these words so that the model fits better the data distribution of Chinese documents, and thus it improves the classification accuracy. The new classifier then re-labels the Chinese documents and the process is repeated for several iterations. Algorithm 2 illustrates this semi-supervised learning process.

Algorithm 2 Semi-supervised learning for cross-lingual text classification

```
$L_s$ ← labeled data in the source language
$U_t$ ← unlabeled data in the target language
$C_s = train(L_s)$
$C_t = translate(C_s)$
repeat
    $Label(U, C_t)$
    $L \leftarrow select(confidence(U, C_t))$
    $C_t \leftarrow train(L)$
until stopping criterion is met
return $C_t$
```

The confidence score associated with the documents is calculated based on the probabilities of the class. For a binary classifier the confidence of classifying the document d is calculated as:

$$confidence(d) = \left|log(\frac{P(c|d)}{P(\overline{c}|d)})\right|$$

An unlabeled document is selected as training data for a new classifier when its confidence score is above a threshold.

5 Experiments and Evaluation

To evaluate the effectiveness of our method, we carry out several experiments. First, we compare the performance of our method on five different categories, from five different domains, in order to see its generality and applicability on different domains. We also run experiments with two different language pairs - English-Chinese and English-French - to see if the distance between language families influences the effectiveness of our method.

To determine the performance of the method with respect to other approaches, we compare the classification accuracy with that of a machine translation approach that translates the training (test) data from the source language to the target language, as well as with a classifier trained on monolingual training data in the target language.

Finally, we evaluate the performance of each of the two steps of our proposed method. First, we evaluate the model translated with the parameters learned with EM, and then the model after the semi-supervised learning for data distribution adaptation with different parameters, including the number of iterations and different amounts of unlabeled data.

5.1 Data Set

Since a standard evaluation benchmark for cross-lingual text classification is not available, we built our own data set from Yahoo! RSS news feeds. The news feed contains news articles from October 1st 2009 to December 31st 2009. We collected a total of 615731 news articles, categorized by their editors into topics such as "sports" or "business". We selected five categories for our experiments, namely "sports", "health", "business", "entertainment", "education". The Yahoo! RSS news feed includes news in many languages, including English, Chinese, French, Spanish, and others.

We experimented on two language pairs, English-Chinese and English-French, selected for their diversity: English and Chinese are disparate languages with very little common vocabulary and syntax, whereas English and French are regarded as more similar. We expect to evaluate the impact of the distance of languages on the effectiveness of our method. In both cases, English is regarded as the source language, where training data are available, and Chinese and French are the target languages for which we want to build text classifiers. Note that regardless of the language, the documents are assigned with one of the five category labels mentioned above. Table 1 shows the distribution of documents across categories and across languages.

Category	English	Chinese	French
sports	23764	14674	18398
health	15627	11769	12745
business	34619	23692	28740
entertainment	26876	21470	23756
education	16488	14353	15753

Table 1: number of documents in each class

Before building the classification model, several preprocessing steps are applied an all the documents. First, the HTML tags are removed, and advertisements and navigational information are also eliminated. For the Chinese corpus, all the Chinese characters with BIG5 encoding are converted into GB2312 and the Chinese texts are segmented into words. For the translation, we use the LDC bilingual dictionary[1] for Chinese English and "stardict" [2] for Spanish English.

5.2 Model Translation

To transfer a model learned in one language to another, we can translate all the bag-of-word features according to a bilingual lexicon. Due to the translation ambiguity of each feature word, we compare three different ways of model translation. One method is to equally assign probabilities to all the translations for a given source language word, and to translate a word we randomly pick a translation from all of its translation candidates. We denote this as "EQUAL" and it is our baseline method. Another way is to calculate the translation probability based on the frequencies of the translation words in the target language itself. For instance, the English word "bush" can be translated into "布什" , "树丛" or "套管" . We can obtain the following unigram counts of these translation words in our Yahoo! RSS news corpus.

count	translation	sense
582	布什	Goerge W. Bush
43	树丛	small trees
2	套管	canula

We can estimate that $P(布什|bush) = 582/(582 + 43 + 2) = 92.8\%$ and so forth. This method often allows us to estimate reasonable translation probabilities and we use "UNIGRAM" to denote this method. And finally the third model translation approach is to use the translation probability learned with the EM algorithm proposed in this paper. The initial parameters of the EM algorithm are set to the probabilities calculated with the "UNIGRAM" method and we use 4000 unlabeled documents in Chinese

[1] http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2002L27

[2] http://stardict.sourceforge.net/Dictionaries.php

to learn translation probabilities with EM. We first train an English classification model for the topic of "sport" and then translate the model into Chinese using translation probabilities estimated by the above three different methods. The three translated models are applied to Chinese test data and we measure the precision, recall and F-score as shown in Table 2.

Method	P	R	F
EQUAL	71.1	70.6	70.8
UNIGRAM	79.5	77.8	78.6
EM	83.1	84.7	83.9

Table 2: Comparison of different methods for model translation

From this table we can see that the baseline method has lowest classification accuracy due to the fact that it is unable to handle translation ambiguity since picking any one of the translation word is equally likely. "UNIGRAM" shows significant improvement over "EQUAL" as the occurrence count of the translation words in the target language can help disambiguate the translations. However occurrence count in a monolingual corpus may not always be the true translation probability. For instance, the English word "work" can be translated into "工作(labor)" and "工厂(factory)" in Chinese. However, in our Chinese monolingual news corpus, the count for "工厂(factory)" is more than that of "工作(labor)" even though "工作(labor)" should be a more likely translation for "work". The "EM" algorithm has the best performance as it is able to learn translation probabilities by looking at documents in both source language and target language instead of just a single language corpus.

5.3 Cross Language Text Classification

To evaluate the effectiveness of our method on cross language text classification, we implement several methods for comparison. In each experiment, we run a separate classification for each class, using a one-versus-all binary classification.

ML (Monolingual). We build a monolingual text classifier by training and testing the text classification system on documents in the same language. This method plays the role of an upper-bound, since the best classification results are expected when monolingual training data is available.

MT (Machine Translation). We use the Systran 5.0 machine translation system to translate the documents from one language into the other in two directions. The first direction translates the training data from the source language into the target language, and then trains a model in the target language. This direction is denoted as **MTS**. The second direction trains a classifier in the source language and translates the test data into the source language. This direction is denoted as **MTT**. In our experiments, Systran generates the single best translation of the text as most off-the-shelf machine translation tools do.

EM (Model Translation with EM). This is the first step of our proposed method. We used 4,000 unlabeled documents to learn translation probabilities with the EM algorithm and the translation probabilities are leveraged to translate the model. The rest of the unlabeled documents are used for other experimental purpose.

SEMI (Adapted Model after Semi-Supervised Learning). This is our proposed method, after both model translation and semi-supervised learning. In the semi-supervised learning, we use 6,000 unlabeled target language documents with three training iterations.

In each experiment, the data consists of 4,000 labeled documents and 1,000 test documents (e.g., in the cross-lingual experiments, we use 4,000 English annotated documents and 1,000 Chinese or French test documents). For a given language, the same test data is used across all experiments.

Table 3 shows the performance of the various classification methods. The **ML** (Monolingual) classifier has the best performance, as it is trained on labeled data in the target language, so that there is no information loss and no distribution discrepancy due to a model translation. The **MT** (machine translation) based approach scores the lowest accuracy, probably because the machine translation software produces only its best translation, which is often error-prone, thus leading to poor classification accuracy. In addition, the direct application of a classification model from one language to an-

Category	English → Chinese														
	ML			MTS			MTT			EM			SEMI		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
sports	96.1	94.3	95.2	80.6	81.7	81.2	81.7	83.8	82.7	83.1	84.7	83.9	92.1	91.8	91.9
health	95.1	93.1	94.1	80.8	81.5	81.2	81.6	83.5	82.6	84.5	85.8	85.2	90.2	91.7	90.9
business	91.6	93.1	92.4	81.3	81.9	81.6	80.7	81.0	80.9	81.6	82.0	81.8	87.3	89.3	88.3
entertainment	88.1	88.3	88.2	76.1	78.8	77.5	75.3	78.9	77.1	76.8	79.7	78.2	83.2	83.8	83.5
education	79.1	82.2	80.6	70.2	72.5	71.8	71.1	72.0	71.6	71.2	73.7	72.5	76.2	79.8	78.0
	English → French														
sports	95.8	95.0	95.4	82.8	83.6	83.2	82.1	83.0	82.5	85.3	87.1	86.2	92.5	92.1	92.3
health	94.2	94.5	94.3	82.6	83.9	83.2	81.8	83.0	82.4	86.2	87.2	86.6	92.0	92.2	92.1
business	90.1	92.2	91.1	81.4	82.1	81.7	81.3	81.8	81.8	84.4	84.3	84.4	88.3	89.2	88.8
entertainment	87.4	87.2	87.3	76.6	79.1	77.8	76.0	78.8	77.4	78.9	81.0	80.0	84.3	85.5	84.9
education	78.8	81.8	80.3	72.1	74.8	73.5	72.3	72.7	72.5	73.8	76.2	75.0	76.3	80.1	78.2

Table 3: Comparison of different methods and different language pairs

other does not adapt to the distribution of the second language, even if the documents belong to the same domain. Comparing the two **MT** alternatives, we can see that translating the training data (**MTS**) has better performance than translating the test data (**MTT**). The reason is that when the model is trained on the translated training data, the model parameters are learned over an entire collection of translated documents, which is less sensitive to translation errors than translating a test document on which the classification is performed individually.

Our **EM** method for translating model features outperforms the machine translation approach, since it does not only rely on the best translation by the machine translation system, but instead takes into account all possible translations with knowledge learned specifically from the target language. Additionally, the **SEMI** (semi-supervised) learning is shown to further improve the classification accuracy. The semi-supervised learning is able to not only help adapt the translated model to fit the words distribution in the target language, but it also compensates the distortion or information loss during the model translation process as it can down-weigh the incorrectly translated features.

The improvement in performance for both the **EM** and the **SEMI** methods is consistent across the five different domains, which indicates that the methods are robust and they are insensitive to the domain of the data.

The performance of the two language pairs English-Chinese and English-French shows a difference as initially hypothesized. In both the **EM** and the **SEMI** models, the classification accuracy of English-French exceeds that of English-Chinese, which is probably explained by the fact that there is less translation ambiguity in similar languages, and they have more similar distributions. Note that the monolingual models in French and Chinese perform comparably, which means the difficulty of the test data is similar between the two target languages.

5.4 Model Adaptation with Semi-Supervised Learning

Finally, to gain further insights into our proposed adaptation method, we run several experiments with different parameters for the semi-supervised learning stage. As these experiments are very time consuming, we run them only on Chinese.

For each of the five categories, we train a classification model using the 4,000 training documents in English and then translate the model into Chinese with the translation parameters learned with **EM** on 20,000 unlabeled Chinese documents. Then we further train the translated model on a set of unlabeled Chinese documents using a different number of iterations and a different amount of unlabeled documents. Figures 1 and 2 show the results of these evaluations.

As the plots show, the use of unlabeled data in the target language can improve the cross-language classification by learning new knowledge in the target language. Larger amounts of unlabeled data in general help, although the marginal benefit drops with increasing amounts of data. Regarding the number of iterations, the best performance is

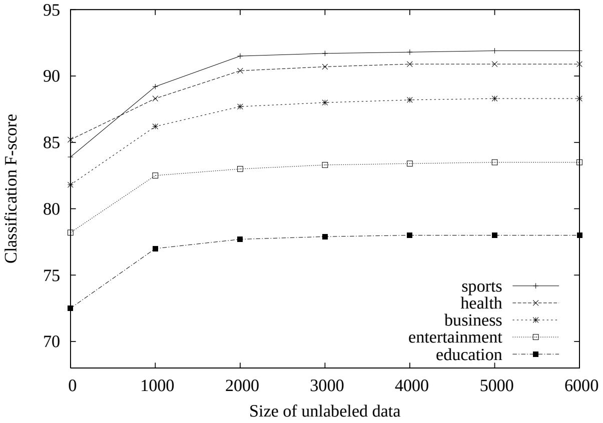


Figure 1: Change in classification F-score for an increasing amount of unlabeled data in the target language

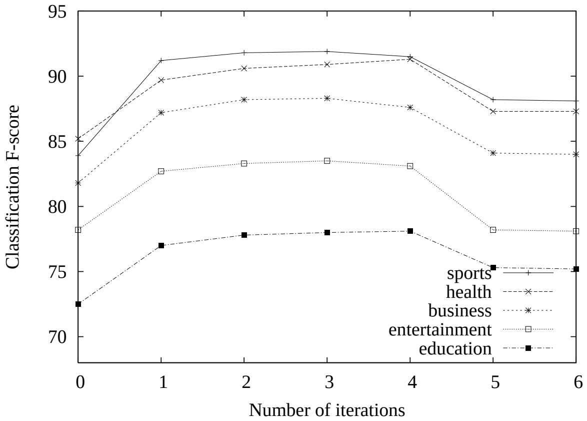


Figure 2: Change in classification F-score for a different number of iterations

achieved after 3-4 iterations.

6 Conclusions

In this paper, we proposed a novel method for cross-lingual text classification. Our method ports a classification model trained in a source language to a target language, with the translation knowledge being learned using the EM algorithm. The model is further tuned to fit the distribution in the target language via semi-supervised learning. Experiments on different datasets covering different languages and different domains show significant improvement over previous methods that rely on machine translation. Moreover, the cross-lingual classification accuracy obtained with our method was found to be close to the one achieved using monolingual text classification.

Acknowledgments

The work of the second author has been partially supported by National Science Foundation awards #0917170 and #0747340. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

S. Argamon, M. Koppel, and G. Avneri. 1998. Style-based text categorization: What newspaper am i reading? In *AAAI-98 Workshop on Learning for Text Categorization*, Madison.

C. Banea, R. Mihalcea, J. Wiebe, and S. Hassan. 2008. Multilingual subjectivity analysis using machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*, Honolulu, Hawaii.

A. Berger, S. Della Pietra, and V. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, June.

P. Brown, S. della Pietra, V. della Pietra, and R. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2).

W. Dai, G. Xue, Q. Yang, and Y. Yu. 2007. Transferring naive bayes classifiers for text classification. In *In Proceedings of the 22nd AAAI Conference on Artificial Intell igence*, pages 540–545.

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 (Methodological)*, 39(1).

B. Fortuna and J. Shawe-Taylor. 2005. The use of machine translation tools for cross-lingual text mining. In *Learning With Multiple Views, Workshop at the 22nd International Conference on Machine Learning (ICML).*

A. Gliozzo and C. Strapparava. 2006. Exploiting comparable corpora and bilingual dictionaries for cross-language text categorization. In *Proceedings of the Conference of the Association for Computational Linguistics*, Sydney, Australia.

R. Hwa, P. Resnik, and A. Weinberg. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*. Special issue on Parallel Texts, editors R. Mihalcea and M. Simard.

T. Joachims. 1997. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, Nashville, US.

T. Joachims. 1998. Text categorization with Support Vector Machines: learning with mny relevant features. In *Proceedings of the European Conference on Machine Learning*, pages 137–142.

P. Koehn and K. Knight. 2000. Estimating word translation probabilities from unrelated monolingua l corpora using the em algorithm. In *National Conference on Artificial Intelligence (AAAI 2000) Lang kilde*, pages 711–715.

A. McCallum and K. Nigam. 1998. A comparison of event models for Naive Bayes text classification. In *Proceedings of AAAI-98 Workshop on Learning for Text Categorization*.

R. Mihalcea, C. Banea, and J. Wiebe. 2007. Learning multilingual subjective language via cross-lingual projections. In *Proceedings of the Association for Computational Linguistics*, Prague, Czech Republic.

C. Monz and B.J. Dorr. 2005. Iterative translation disambiguation for cross-language information retrieval. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Salvador, Brazil.

A. Moschitti. 2003. A study on optimal paramter tuning for Rocchio text classifier. In *Proceedings of the European Conference on Information Retrieval*, Italy.

H.T. Ng, B. Wang, and Y.S. Chan. 2003. Exploiting parallel texts for word sense disambiguation: An empirical study. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)*, Sapporo, Japan, July.

J.-Y. Nie, M. Simard, P. Isabelle, and R. Durand. 1999. Cross-language information retrieval based on parallel texts and automatic mining of parallel texts from the Web. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*.

K. Nigam and R. Ghani. 2000. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the Conference on Information and Knowledge Management (CIKM 2000)*, McLean, VA, November.

K. Nigam, J. Lafferty, and A. McCallum. 1999. Using maximum entropy for text classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*.

J.S. Olsson, D. W. Oard, and J. Hajic. 2005. Cross-language text classification. In *Proceedings of the 28th Annual international ACM SIGIR Conference on Research and Development in information Retrieval*.

B. Pang and L. Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, Barcelona, Spain, July.

M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. 1998. A Bayesian approach to filtering junk e-mail. In *AAAI-98 Workshop on Learning for Text Categorization*, Madison.

J. Schler, M. Koppel, S. Argamon, and J. Pennebaker. 2006. Effects of age and gender on blogging. In *Proceedings of 2006 AAAI Spring Symposium on Computational Approaches for Analyzing Weblogs*, pages 199–204, Stanford.

L. Shi, C. Niu, M. Zhou, and J. Gao. 2006. A dom tree alignment model for mining parallel data from the web. In *Proceedings of the Annual Meeting of the Association for Computational Lingusitics (ACL 2006)*, Sydney, Australia.

V. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer, New York.

X. Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the Association of Computational Linguistics and the International Joint Conference on Natural Language Processing*, Singapore, August.

SRL-based Verb Selection for ESL

[1,2]Xiaohua Liu, [3]Bo Han[*], [4]Kuan Li[*], [5]Stephan Hyeonjun Stiller and [2]Ming Zhou
[1]School of Computer Science and Technology
Harbin Institute of Technology
[2]Microsoft Research Asia
[3]Department of Computer Science and Software Engineering
The University of Melbourne
[4]College of Computer Science
Chongqing University
[5]Computer Science Department
Stanford University
{xiaoliu, mingzhou, v-kuli}@microsoft.com
b.han@pgrad.unimelb.edu.au
sstiller@stanford.edu

Abstract

In this paper we develop an approach to tackle the problem of verb selection for learners of English as a second language (ESL) by using features from the output of Semantic Role Labeling (SRL). Unlike existing approaches to verb selection that use local features such as n-grams, our approach exploits semantic features which explicitly model the usage context of the verb. The verb choice highly depends on its usage context which is not consistently captured by local features. We then combine these semantic features with other local features under the generalized perceptron learning framework. Experiments on both in-domain and out-of-domain corpora show that our approach outperforms the baseline and achieves state-of-the-art performance.

1 Introduction

Verbs in English convey actions or states of being. In addition, they also communicate sentiments and imply circumstances, e.g., in "*He got [gained] the scholarship after three interviews.*", the verb "*gained*" may indicate that the "*scholarship*" was competitive and required the agent's efforts; in contrast, "*got*" sounds neutral and less descriptive. Since verbs carry multiple important functions, misusing them can be misleading, e.g., the native speaker could be confused when reading "*I like looking [reading] books*". Unfortunately, according to (Gui and Yang, 2002; Yi et al., 2008), more than 30% of the errors in the Chinese Learner English Corpus (CLEC) are verb choice errors. Hence, it is useful to develop an approach to automatically detect and correct verb selection errors made by ESL learners.

However, verb selection is a challenging task because verbs often exhibit a variety of usages and each usage depends on a particular context, which can hardly be adequately described by conventional n-gram features. For instance, both "*made*" and "*received*" can complete "*I have __ a telephone call.*", where the usage context can be represented as "*made/received a telephone call*"; however, in "*I have __ a telephone call from my boss*", the prepositional phrase "*from my boss"* becomes a critical part of the context, which now cannot be described by n-gram features, resulting in only "*received*" being suitable.

Some researchers (Tetreault and Chodorow, 2008) exploited syntactic information and n-gram features to represent verb usage context. Yi et al. (2008) introduced an unsupervised web-based proofing method for correcting verb-noun collocation errors. Brockett et al. (2006) employed phrasal Statistical Machine Translation (SMT) techniques to correct countability errors. None of their methods incorporated semantic information.

[*] This work has been done while the author was visiting Microsoft Research Asia.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1068–1076,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

Unlike the other papers, we derive features from the output of an SRL (Màrquez, 2009) system to explicitly model verb usage context. SRL is generally understood as the task of identifying the arguments of a given verb and assigning them semantic labels describing the roles they play. For example, given a sentence "*I want to watch TV tonight*" and the target predicate "*watch*", the output of SRL will be something like "*I [A0] want to watch [target predicate] TV [A1] tonight [AM-TMP].*", meaning that the action "*watch*" is conducted by the agent "*I*", on the patient "*TV*", and the action happens "*tonight*".

We believe that SRL results are excellent features for characterizing verb usage context for three reasons: (i) Intuitively, the predicate-argument structures generated by SRL systems capture major relationships between a verb and its contextual participants and consequently largely determine whether or not the verb usage is proper. For example, in "*I want to watch a match tonight.*", "*match*" is the patient of "*watch*", and "*watch ... match*" forms a collocation, suggesting "*watch"* is appropriately used. (ii) Predicate-argument structures abstract away syntactic differences in sentences with similar meanings, and therefore can potentially filter out lots of noise from the usage context. For example, consider "*I want to watch a football match on TV tonight*": if "*match*" is successfully identified as the agent of "*watch*", "*watch ... football*", which is unrelated to the usage of "*watch*" in this case, can be easily excluded from the usage context. (iii) Research on SRL has made great achievements, including human-annotated training corpora and state-of-the-art systems, which can be directly leveraged.

Taking an English sentence as input, our method first generates correction candidates by replacing each verb with verbs in its pre-defined confusion set; then for every candidate, it extracts SRL-derived features; finally our method scores every candidate using a linear function trained by the generalized perceptron learning algorithm (Collins, 2002) and selects the best candidate as output.

Experimental results show that SRL-derived features are effective in verb selection, but we also observe that noise in SRL output adversely increases feature space dimensions and the number of false suggestions. To alleviate this issue, we use local features, e.g., n-gram-related features, and achieve state-of-the-art performance when all features are integrated.

Our contributions can be summarized as follows:

1. We propose to exploit SRL-derived features to explicitly model verb usage context.
2. We propose to use the generalized perceptron framework to integrate SRL-derived (and other) features and achieve state-of-the-art performance on both in-domain and out-of-domain test sets.

Our paper is organized as follows: In the next section, we introduce related work. In Section 3, we describe our method. Experimental results and analysis on both in-domain and out-of-domain corpora are presented in Section 4. Finally, we conclude our paper with a discussion of future work in Section 5.

2 Related Work

SRL results are used in various tasks. Moldovan et al. (2004) classify the semantic relations of noun phrases based on SRL. Ye and Baldwin (2006) apply semantic role–related information to verb sense disambiguation. Narayanan and Harabagiu (2004) use semantic role structures for question answering. Surdeanu et al. (2003) employ predicate-argument structures for information extraction.

However, in the context of ESL error detection and correction, little study has been carried out on clearly exploiting semantic information. Brockett et al. (2006) propose the use of the phrasal statistical machine translation (SMT) technique to identify and correct ESL errors. They devise several heuristic rules to generate synthetic data from a high-quality newswire corpus and then use the synthetic data together with their original counterparts for SMT training. The SMT approach on the artificial data set achieves encouraging results for correcting countability errors. Yi et al. (2008) use web frequency counts to identify and correct determiner and verb-noun collocation errors. Compared with these methods, our approach explicitly models verb usage context by leveraging the SRL output. The SRL-based semantic features are integrated, along with the local features, into the generalized perceptron model.

3 Our Approach

Our method can be regarded as a pipeline consisting of three steps. Given as input an English sentence written by ESL learners, the system first checks every verb and generates correction candidates by replacing each verb with its confusion set. Then a feature vector that represents verb usage context is derived from the outputs of an SRL system and then multiplied with the feature weight vector trained by the generalized perceptron. Finally, the candidate with the highest score is selected as the output.

3.1 Formulation

We formulate the task as a process of generating and then selecting correction candidates:

$$s^* = \arg\max_{s \in GEN(s')} Score(s) \qquad (1)$$

Here s' denotes the input sentence for proofing, $GEN(s')$ is the set of correction candidates, and $Score(s)$ is the linear model trained by the perceptron learning algorithm, which will be discussed in section 3.4.

We call every target verb in s' a *checkpoint*. For example, "*sees*" is a checkpoint in "*Jane **sees** TV every day.*". Correction candidates are generated by replacing each checkpoint with its confusions. Table 1 shows a sentence with one checkpoint and the corresponding correction candidates.

Input	*Jane **sees** TV every day.*
Candidates	*Jane **watches** TV every day.*
	*Jane **looks** TV every day.*
	…

Table 1. Correction candidate list.

One state-of-the-art SRL system (Riedel and Meza-Ruiz, 2008) is then utilized to extract predicate-argument structures for each verb in the input, as illustrated in Table 2.

Semantic features are generated by combining the predicate with each of its arguments; e.g., "*watches_A0_Jane*", "*sees_A0_Jane*", "*watches_A1_TV*" and "*sees_A1_TV*" are semantic features derived from the semantic roles listed in Table 2.

Sentence	Semantic roles

*Jane **sees** TV every day*	*Predicate: sees;*
	A0: Jane;
	A1: TV;

*Jane **watches** TV every day*	*Predicate: watches;*
	A0: Jane;
	A1: TV;

Table 2. Examples of SRL outputs.

At the training stage, each sentence is labeled by the SRL system. Each correction candidate s is represented as a feature vector $\Phi(s) \in R^d$, where d is the total number of features. The feature weight vector is denoted as $\vec{w} \in R^d$, and $Score(s)$ is computed as follows:

$$Score(s) = \Phi(s) \cdot \vec{w} \qquad (2)$$

Finally, $Score(s)$ is applied to each candidate, and s^*, the one with the highest score, is selected as the output, as shown in Table 3.

	Correction candidate	Score
s^*	*Jane **watches** TV every day.*	10.8
	*Jane **looks** TV every day.*	0.8
	*Jane **reads** TV every day.*	0.2
	…	…

Table 3. Correction candidate scoring.

In the above framework, the basic idea is to generate correction candidates with the help of predefined confusion sets and apply the global linear model to each candidate to compute the degree of its fitness to the usage context that is represented as features derived from SRL results.

To make our idea practical, we need to solve the following three subtasks: (i) generating the confusion set that includes possible replacements for a given verb; (ii) representing the context with semantic features and other complementary features; and (iii) training the feature weight. We will describe our solutions to those subtasks in the rest of this section.

I	*have*	***made[opened]***	*an*	*American*	*bank*	*account*	*in*	*Boston*	*.*
[A0]		[Predicate]				[A1]	[AM-LOC]		

Table 4. An example of SRL output.

3.2 Generation of Verb Confusion Sets

Verb confusion sets are used to generate correction candidates. Due to the great number of verbs and their diversified usages, manually collecting all verb confusions in all scenarios is prohibitively time-consuming. To focus on the study of the effectiveness of semantic role features, we restrict our research scope to correcting verb selection errors made by Chinese ESL learners and select fifty representative verbs which are among the most frequent ones and account for more than 50% of ESL verb errors in the CLEC data set. For every selected verb we manually compile a confusion set using the following data sources:

1. Encarta treasures. We extract all the synonyms of verbs from the Microsoft Encarta Dictionary, and this forms the major source for our confusion sets.

2. English-Chinese Dictionaries. ESL learners may get interference from their mother tongue (Liu et al., 2000). For example, some Chinese people mistakenly say “see newspaper”, partially because the translation of “see” co-occurs with “newspaper” in Chinese. Therefore English verbs in the dictionary sharing more than two Chinese meanings are collected. For example, “*see*” and “*read*” are in a confusion set because they share the meanings of both “*看*” (“*to see*”*,* “*to read*”) and “*领会*” (“*to grasp*”) in Chinese.

3. An SMT translation table. We extract paraphrasing verb expressions from a phrasal SMT translation table learnt from parallel corpora (Och and Ney, 2004). This may help us use the implicit semantics of verbs that SMT can capture but a dictionary cannot, such as the fact that the verb

Note that verbs in any confusion set that we are not interested in are dropped, and that the verb itself is included in its own confusion set. We leave it to our future work to automatically construct verb confusions.

3.3 Verb Usage Context Features

The verb usage context[1] refers to its surrounding text, which influences the way one understands the expression. Intuitively, verb usage context can take the form of a collocation, e.g., “*watch … TV*” in “*I saw [watched] TV yesterday.*” ; it can also simply be idioms, e.g., we say “*kick one’s habit*” instead of “*remove one’s habit*”.

We use features derived from the SRL output to represent verb usage context. The SRL system accepts a sentence as input and outputs all arguments and the semantic roles they play for every verb in the sentence. For instance, given the sentence “*I have opened an American bank account in Boston.*” and the predicate “*opened*”, the output of SRL is listed in Table 4, where *A0* and *A1* are two core roles, representing the agent and patient of an action, respectively, and other roles starting with “*AM-*”are adjunct roles, e.g., *AM-LOC* indicates the location of an action. Predicate-argument structures keep the key participants of a given verb while dropping other unrelated words from its usage context. For instance, in “*My teacher said Chinese is not easy to learn.*”, the SRL system recognizes that “*Chinese*” is not the *A1*-argument of “*said*”. So “*say _ Chinese*”, which is irrelevant to the usage of *said*, is not extracted as a feature.

The SRL system, however, may output erroneous predicate-argument structures, which negatively affect the performance of verb selection. For instance, for the sentence “*He hasn’t done anything but take [make] a lot of money*”, “*lot*” is incorrectly identified as the patient of “*take*”, making it hard to select “*make*” as the proper verb even though “*make money*” forms a sound collocation. To tackle this issue, we use local textual features, namely features related to n-gram, chunk and chunk headword, as shown in Table 5. Back-off features are generated by replacing the word with its POS tag to alleviate data sparseness.

[1] http://en.wikipedia.org/wiki/Context_(language_use)

Local: trigrams
*have_**opened***
*have_**opened**_a*
***opened**_an_American*
*PRP_VBP_**opened***
*VBP_**opened**_DT*
***opened**_DT_JJ*
Local: chunk
*have_**opened***
***opened**_an_American_investment_bank_account*
*PRP_**opened***
***opened**_NN*
Semantic: SRL derived features
*A0_I_**opened***
***opened**_A1_account*
***opened**_AM-LOC_in*
...

Table 5. An example of feature set.

3.4 Perceptron Learning

We choose the generalized perceptron algorithm as our training method because of its easy implementation and its capability of incorporating various features. However, there are still two concerns about this perceptron learning approach: its ineffectiveness in dealing with inseparable samples and its ignorance of weight normalization that potentially limits its ability to generalize. In section 4.4 we show that the training error rate drops significantly to a very low level after several rounds of training, suggesting that the correct candidates can almost be separated from others. We also observe that our method performs well on an out-of-domain test corpus, indicating the good generalization ability of this method. We leave it to our future work to replace perceptron learning with other models like Support Vector Machines (Vapnik, 1995).

In Figure 1, s_i is the *i*th correct sentence within the training data. *T* and *N* represent the number of training iterations and training examples, respectively. $GEN(s^i)$ is the function that outputs all the possible corrections for the input sentence s^i with each *checkpoint* substituted by one of its confusions, as described in Section 3.1. We observe that the generated candidates sometimes contain reasonable outputs for the verb selection task, which should be removed. For instance, in "... *reporters could not take [make] notes or tape the conversation*", both "*take*" and "*make*" are suitable verbs in this context. To fix this issue, we trained a trigram language model using SRILM (Stolcke, 2002) on LDC data[2] , and calculated the logarithms of the language model score for the original sentence and its artificial manipulations. We only kept manipulations with a language model score that is *t* lower than that of the original sentence. We experimentally set *t* = *5*.

```
Inputs: training examples $s^i$, $i=1...N$
Initialization: $\vec{w}=0$
Algorithm:
  For $r= 1.. T$, $i= 1..N$
  Calculate $o=\arg\max_{s=Gen(s^i)} \Phi(s)\cdot\vec{w}$
  If $s^i \neq o$
      $\vec{w}=\vec{w}+\Phi(s^o)-\Phi(o)$
Outputs: $\vec{w}$
```

Figure 1. The perceptron algorithm, adapted from Collins (2002).

Φ in Figure 1 is the feature extraction function. $\Phi(o)$ and $\Phi(s^i)$ are vectors extracted from the output and oracle, respectively. A vector field is filled with 1 if the corresponding feature exists, or 0 otherwise; $\vec{w}$ is the feature weight vector, where positive elements suggest that the corresponding features support the hypothesis that the candidate is correct.

The training process is to update $\vec{w}$, when the output differs from the oracle. For example, when o is "*I want to **look** TV*" and s^i is "*I want to **watch** TV*", $\vec{w}$ will be updated.

We use the averaged Perceptron algorithm (Collins, 2002) to alleviate overfitting on the training data. The averaged perceptron weight vector is defined as

$$\vec{\gamma}=\frac{1}{TN}\sum_{\mathrm{i}=1..N,r=1..T}\vec{w}^{i,r} \qquad (3)$$

where $\vec{w}^{i,r}$ is the weight vector immediately after the *i*th sentence in the *r*th iteration.

[2] http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp ?catalogId=LDC2005T12

4 Experiments

In this section, we compare our approach with the SMT-based approach. Furthermore, we study the contribution of predicate-argument-related features, and the performances on verbs with varying distance to their arguments.

4.1 Experiment Preparation

The training corpus for perceptron learning was taken from LDC2005T12. We randomly selected newswires containing target verbs from the *New York Times* as the training data. We then used the *OpenNLP* package[3] to extract sentences from the newswire text and to parse them into the corresponding tokens, POS tags, and chunks. The SRL system is built according to Riedel and Meza-Ruiz (2008), using the CoNLL-2008 shared task data for training. We assume that the newswire data is of high quality and free of linguistic errors, and finally we gathered 20000 sentences that contain any of the target verbs we were focusing on. We experimentally set the number of training rounds to $T = 50$.

We constructed two sets of testing data for in-domain and out-of-domain test purposes, respectively. To construct the in-domain test data, we first collected all the sentences that contain any of the verbs we were interested in from the previous unused LDC dataset; then we replaced any target verb in our list with a verb in its confusion set; next, we used the language-model-based pruning strategy described in 3.4 to drop possibly correct manipulations from the test data; and finally we randomly sampled 5000 sentences for testing.

To build the out-of-domain test dataset, we gathered 186 samples that contained errors related to the verbs we were interested in from English blogs written by Chinese and from the CLEC corpus, which were then corrected by an English native speaker. Furthermore, for every error involving the verbs in our target list, both the verb and the word that determines the error are marked by the English native speaker.

4.2 Baseline

We built up a phrasal SMT system with the word re-ordering feature disabled, since our task only concerns the substitution of the target verb. To construct the training corpus, we followed the idea in Brockett et al. (2006), and applied a similar strategy described in section 3.4 to the SRL system's training data to generate aligned pairs.

4.3 Evaluation Metric

We employed the following metrics adapted from (Yi et al., 2008): *revised precision (RP), recall of the correction (RC)* and *false alarm (FA).*

$$RP = \frac{\#\,of\ Correct\ Proofings}{\#\,of\ All\ Checkpoints} \tag{4}$$

RP reflects how many outputs are correct usages. The output is regarded as a correct suggestion if and only if it is exactly the same as the answer. Paraphrasing scenarios, for example, the case that the output is "*take notes*" and the answer is "*make notes*", are counted as errors.

$$RC = \frac{\#\,of\ Correct\ Modified\ Proofings}{\#\,of\ Total\ Errors} \tag{5}$$

RC indicates how many erroneous sentences are corrected among all the errors. It measures the system's coverage of verb selection errors.

$$FA = \frac{\#\,of\ Incorrect\ Modified\ Checkpoints}{\#\,of\ All\ Checkpoints} \tag{6}$$

FA is related to the cases where a correct verb is mistakenly replaced by an inappropriate one. These false suggestions are likely to disturb or even annoy users, and thus should be avoided as much as possible.

4.4 Results and Analysis

The training error curves of perceptron learning with different feature sets are shown in Figure 2. They drop to a low error rate and then stabilize after a few number of training rounds, indicating that most of the cases are linearly separable and that perceptron learning is applicable to the verb selection task.

We conducted feature selection by dropping features that occur less than *N* times. Here *N* was experimentally set to 5. We observe that, after feature selection, some useful features such as "*watch_A1_TV*" and "*see_A1_TV*" were kept, but some noisy features like *"Jane_A0_sees"* and *"Jane_A0_watches"* were removed, suggesting the effectiveness of this feature selection approach.

[3] http://opennlp.sourceforge.net/

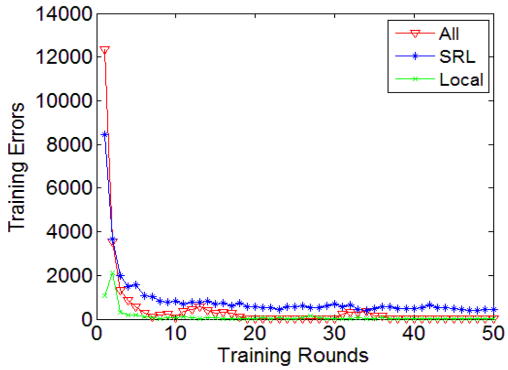


Figure 2. Training error curves of the perceptron.

We tested the baseline and our approach on the in-domain and out-of-domain corpora. The results are shown in Table 7 and 8, respectively.

In the in-domain test, the SMT-based approach has the highest false alarm rate, though its output with word insertions or deletions is not considered wrong if the substituted verb is correct. Our approach, regardless of what feature sets are used, outperforms the SMT-based approach in terms of all metrics, showing the effectiveness of perceptron learning for the verb selection task. Under the perceptron learning framework, we can see that the system using only SRL-related features has higher revised precision and recall of correction, but also a slightly higher false alarm rate than the system based on only local features. When local features and SRL-derived features are integrated together, the state-of-the-art performance is achieved with a 5% increase in recall, and minor changes in precision and false alarm.

In the out-of-domain test, the SMT-based approach performs much better than in the in-domain test, especially in terms of false alarm rate, indicating the SMT-based approach may favor short sentences. However, its recall drops greatly. We observe similar performance differences between the systems with different feature sets under the same perceptron learning framework, reaffirming the usefulness of the SRL-based features for verb selection.

We also conducted significance test. The results confirm that the improvements (SRL+Local vs. SMT-based) are statistically significant (p-value < 0.001) for both the open-domain and the in-domain experiments.

Furthermore, we studied the performance of our system on verbs with varying distance to their arguments on the out-of-domain test corpus.

Local	d<=2	2<d<=4	d>4
RP	64.3%	60.3%	59.4%
RC	34.6%	33.1%	28.9%
FA	3.0%	6.3%	5.0%
SRL	d<=2	2<d<=4	d>4
RP	65.1%	60.1%	62.1%
RC	40.3%	34.0%	36.9%
FA	5.0%	6.7%	6.3%

Table 9. Performance on verbs with different distance to their arguments on out-of-domain test data.

Table 9 shows that the system with only SRL-derived features performs significantly better than the system with only local features on the verb whose usage depends on a distant argument, i.e., one where the number of words between the predicate and the argument is larger than 4. To understand the reason, consider the following sentence:

"*It's raining outside. Please wear[take] the black raincoat with you.*"

	SMT-based	Our method		
		SRL	Local	SRL + Local
RP	48.4%	64.5%	62.2%	66.4%
RC	23.5%	40.2%	32.9%	46.4%
FA	13.3%	5.6%	4.2%	6.8%

Table 7. In-domain test results.

	SMT-based	Our method		
		SRL	Local	SRL + Local
RP	50.7%	64.0%	62.6%	65.5%
RC	13.5%	39.0%	33.3%	44.0%
FA	6.1%	5.5%	4.0%	6.5%

Table 8. Out-of-domain test results.

Intuitively, "*wear*" and "*take*" seem to fill the blank well, since they both form a collocation with "*raincoat*"*;* however, when *"with [AM-MNR] you*" is considered as part of the context, "*wear*" no longer fits it and "*take*" wins. In this case, the long-distance feature devised from *AM-MNR* helps select the suitable verb, while the trigram features cannot because they cannot represent the long distance verb usage context.

We also find some typical cases that are beyond the reach of the SRL-derived features. For instance, consider "*Everyone doubts [suspects] that Tom is a spy.*". Both of the verbs can be followed by a clause. However, the SRL system regards "*is*", the predicate of the clause, as the patient, resulting in features like "*doubt_A1_is*" and "*suspect_A1_is*", which capture nothing about verb usage context. However, if we consider the whole clause "*suspect_Tom is a spy*" as the patient, this could result in a very sparse feature that would be filtered. In the future, we will combine word-level and phrase-level SRL systems to address this problem.

Besides its incapability of handling verb selection errors involving clauses, the SRL-derived features fail to work when verb selection depends on deep meanings that cannot be captured by current shallow predicate-argument structures. For example, in "*He was wandering in the park, spending [killing] his time watching the children playing.*", though "*spending*" and "*killing*" fit the syntactic structure and collocation agreement, and express the meaning "*to allocate some time doing something*", the word "*wandering*" suggests that "*killing*" may be more appropriate. Current SRL systems cannot represent the semantic connection between two predicates and thus are helpless for this case. We argue that the performance of our system can be improved along with the progress of SRL.

5 Conclusions and Future Work

Verb selection is challenging because verb usage highly depends on the usage context, which is hard to capture and represent. In this paper, we propose to utilize the output of an SRL system to explicitly model verb usage context. We also propose to use the generalized perceptron learning framework to integrate SRL-derived features with other features. Experimental results show that our method outperforms the SMT-based system and achieves state-of-the-art performance when SRL-related features and other local features are integrated. We also show that, for cases where the particular verb usage mainly depends on its distant arguments, a system with only SRL-derived features performs much better than the system with only local features.

In the future, we plan to automatically construct confusion sets, expand our approach to more verbs and test our approach on a larger size of real data. We will try to combine the outputs of several SRL systems to make our system more robust. We also plan to further validate the effectiveness of the SRL-derived features under other learning methods like SVMs.

Acknowledgment

We thank the anonymous reviewers for their valuable comments. We also thank Changning Huang, Yunbo Cao, Dongdong Zhang, Henry Li and Mu Li for helpful discussions.

References

Francis Bond, Kentaro Ogura, and Satoru Ikehara. 1994. Countability and number in Japanese to English machine translation. *Proc. of the 15th conference on Computational Linguistics*, pages 32-38.

Chris Brockett, William B. Dolan, and Michael Gamon. 2006. Correcting ESL errors using phrasal SMT techniques. *Proc. of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting on Association for Computational Linguistics*, pages 249-256.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. *Proc. of the ACL-02 Conference on Empirical Methods in Natural Language Processing*, pages 1-8.

Jens Eeg-Olofsson and Ola Knutsson. 2003. Automatic Grammar Checking for Second Language Learners – the Use of Prepositions. *Proc. of NoDaliDa*.

Michael Gamon, Jianfeng Gao, Chris Brockett, Alexandre Klementiev, William B. Dolan, Dmitrtiy Belenko, and Lucy Vanderwende. 2008. Using Contextual Speller Techniques and Language Modeling for ESL Error Correction. *Proc. of the International Joint Conference on Natural Language Processing*.

Shichun Gui and Huizhong Yang. 2002. *Chinese Learner English Corpus*. Shanghai Foreign Languages Education Press, Shanghai, China.

Julia E. Heine. 1998. Definiteness predictions for Japanese noun phrases. *Proc. of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 519-525.

John Lee and Stephanie Seneff. 2008. Correcting misuse of verb forms. *Proc. of the 46th Annual Meeting on Association for Computational Linguistics,* pages 174-182.

Ting Liu, Ming Zhou, Jianfeng Gao, Endong Xun and Changning Huang. 2000. PENS: A Machine-aided English Writing System for Chinese Users. *Proc. of the 38th Annual Meeting on Association for Computational Linguistics*, pages 529-536.

Lluís Màrquez. 2009. *Semantic Role Labeling Past, Present and Future*, Tutorial of ACL-IJCNLP 2009.

Dan Moldovan, Adriana Badulescu, Marta Tatu, Daniel Antohe and Roxana Girju. 2004. Models for the semantic classification of noun phrases. *Proc. of the HLT-NAACL Workshop on Computational Lexical Semantics*, pages 60-67.

Srini Narayanan and Sanda Harabagiu. 2004. Question answering based on semantic structures. *Proc. of the 20th International Conference on Computational Linguistics*, pages 693-701.

Franz J. Och and Hermann Ney. 2004. The Alignment Template Approach to Statistical Machine Translation. *Journal of Computational Linguistics,* 30(4), pages 417-449.

Sebastian Riedel and Ivan Meza-Ruiz. 2008. Collective semantic role labelling with Markov Logic. *Proc. of the Twelfth Conference on Computational Natural Language Learning*, pages 193-197.

Andreas Stolcke. 2002. SRILM -- An Extensible Language Modeling Toolkit. *Proc. of International Conference on Spoken Language Processing*, pages: 901-904.

Mihai Surdeanu, Lluis Màrquez, Xavier Carreras, and Pere R. Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research*, page 105-151.

Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. *Proc. of the 41st Annual Meeting on Association for Computational Linguistics*, pages 8-15.

Joel R. Tetreault and Martin Chodorow. 2008. The ups and downs of preposition error detection in ESL writing. *Proc. of the 22nd international Conference on Computational Linguistics*, pages 865-872.

Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.

Patrick Ye and Timothy Baldwin. 2006. Verb Sense Disambiguation Using Selectional Preferences Extracted with a State-of-the-art Semantic Role Labeler. *Proc. of the Australasian Language Technology Workshop*, pages 141-148.

Xing Yi, Jianfeng Gao, and William B. Dolan. 2008. A Web-based English Proofing System for English as a Second Language Users. *Proc. of International Joint Conference on Natural Language Processing*, pages 619-624.

Context Comparison of Bursty Events in Web Search and Online Media

Yunliang Jiang
University of Illinois
Urbana, IL, 61801
jiang8@illinois.edu

Cindy Xide Lin
University of Illinois
Urbana, IL, 61801
xidelin2@illinois.edu

Qiaozhu Mei
University of Michigan
Ann Arbor, MI, 48109
qmei@umich.edu

Abstract

In this paper, we conducted a systematic comparative analysis of language in different contexts of bursty topics, including web search, news media, blogging, and social bookmarking. We analyze (1) the content similarity and predictability between contexts, (2) the coverage of search content by each context, and (3) the intrinsic coherence of information in each context. Our experiments show that social bookmarking is a better predictor to the bursty search queries, but news media and social blogging media have a much more compelling coverage. This comparison provides insights on how the search behaviors and social information sharing behaviors of users are correlated to the professional news media in the context of bursty events.

1 Introduction

Search is easy. Every day people are repeating the queries they have used before, trying to access the same web pages. A smart search engine tracks the preference and returns it next time when it sees the same query. When I search for "msr" I always try to access *Microsoft research*; and even if I misspelled it, a smart search engine could suggest a correct query based on my query history, the current session of queries, and/or the queries that other people have been using.

Search is hard. I search for "social computing" because there was such a new program in NSF; but the search engine might have not yet noticed that. People use "msg" to access *monosodium glutamate* in most of the cases, but tonight there is a big game in *Madison square garden*. *H1N1* suddenly became a hot topic, followed by a burst of the rumor that *it was a hoax*, and then the *vaccine*. The information need of users changed dramatically during such a period. When a new event happens, the burst of new contents and new interests make it hard to predict what people would search and to suggest what queries they should use.

Web search is easy when the information need of the users is stable and when we have enough historical clicks. It becomes much more difficult when a new information need knocks the door or when there is a sudden change of the information need. Such a shift of the information need is usually caused by a burst of new events or new interests.

When we are lack of enough historical observations, why don't we seek help from other sources? A bursting event will not only influence what we search, but hopefully also affect *what we read*, *what we write*, and *what we tag*. Indeed, there is already considerable effort in seeking help from these sources, by the integration of news and blogs into search results or the use of social bookmarks to enhance search. These conclusions, however, are mostly drawn in a general context (e.g., with general search queries). To what extent are they useful when dealing with busty events? How is the bursting content in web search, news media, social media, and social bookmarks correlating and different from each other? Prior to the development of desirable applications (e.g. enhancing search results, query suggestion, keyword bidding on advertisement, etc) by integrating the information from all these sources, it is appealing to have an investigation of feasibility.

In this work, we conduct a systematic comparative study of what we search, what we read, what

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1077–1087,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

we write, and what we tag in the scenarios of *bursty events*. Specifically, we analyze the language used in different *contexts* of bursty events, including two different query log contexts, two news media contexts, two blog contexts, and an additional context of social bookmarks. A variety of experiments have been conducted, including the content similarity and cross-entropy between sources, the coverage of search queries in online media, and an in-depth semantic comparison of sources based on language networks.

In the rest of this paper, a summary of related work is briefly described in Section 2. We then present the experiments setup in Section 3, The results of the experiments is presented in Section 4. Finally, our major findings from the comparative analysis are drawn in Section 5.

2 Related Work

Recently, a rich body of work has focused on how to find the bursting patterns from time-series data using various approaches such as time-graph analysis (Kleinberg, 2003; Kuman et al., 2003), context-based analysis (Gabrilovich et al., 2004), moving-average analysis (Vlachos et al., 2004), and frequency analysis (Gruhl et al., 2005), etc. These methods are all related to the preprocessing step of our analysis: detecting bursty queries from the query log effectively.

The comparison of *two* web sources at a time is widely studied recently. (Sood et al., 2007) discussed how to leverage the relation between social tags and web blogs. (Lloyd et al., 2006; Gamon et al., 2008; Cointet et al., 2008) investigated the relations between news and blogs. Also some work has aimed to utilize one external web source to help web search. For example, (Diaz, 2009) integrated the news results into general search. (Bao et al., 2007; Heymann et al., 2008; Krause et al., 2008; Bischoff et al., 2008) focused on improving search by the social tags. Compared with the above, our comparison analysis tries to explore the interactions among *multiple* web sources including the search logs.

Similar to our work, some recent work (Adar et al., 2007; Sun et al., 2008) has addressed the comparison among multiple web sources. For example, (Adar et al., 2007) did a comprehensive correlation study among queries, blogs, news and TV results. However, different from the *content-free* analysis above, our work compares the sources based on the *content*.

Our work can lead to many useful search applications, such as query suggestion which takes as input a specific query and returns as output one or several suggested queries. The approaches include query term cooccurrence (Jones et al., 2006), query sessions (Radlinski and Joachims, 2005), and click-through (Mei et al., 2008), respectively.

3 Analysis Setup

Tasks of web information retrieval such as web search generally perform very well on frequent and *navigational* queries (Broder, 2002) such like “chicago” or “yahoo movies.” A considerable challenge in web search remains in how to handle *informational* queries, especially queries that reflect *new* information need and *suddenly changed* information need of users. Many such scenarios are caused by the emergence of bursty events (e.g., “van gogh” became a hot query in May 2006 since a Van Goghs portrait was sold for 40.3 million in New York during that time). The focus of this paper is to analyze how other online media sources react to those bursty events and how those reactions compare to the reaction in web search. This analysis thus serves as an primitive investigation of the feasibility of leveraging other sources to enhance the search of bursty topics.

Therefore, we focus on the “event-related” topics which present as bursty queries submitted to a search engine. These queries not only reflect the suddenly changed information need of users, but also trigger the correlated reactions in other online sources, such as news media, blog media, social bookmarks, etc. We begin with the extraction of bursty topics from the query log.

3.1 Bursty Topic Extraction

Search engine logs (or query logs) store the history of users’ search behaviors, which reflect users’ interests and information need. The query log of a commercial search engine consists of a huge amount of search records, each of which typically contains the following information: the query submitted by a user, the time at which the query was submitted, and/or the URL which the user clicked on after the query was submitted, etc. It is common practice to segment query log into search sessions, each of which represents one user's searching activities in a short period of time.

We explore a sample of the log of the Microsoft Live search engine[1], which contains $14.9M$ search records over 1 month (May 2006).

3.1.1 Find bursty queries from query log

How to extract the queries that represent bursty events? We believe that bursty queries present the pattern that its day-by-day search volume shows a significant spike – that is, the frequency that the user submit this query should suddenly increase at one specific time and drop down after a while. This assumption is consistent with existing work of finding bursty patterns in emails, scientific literature (Kleinberg, 2003), and blogs (Gruhl et al., 2005).

Following (Gruhl et al., 2005), we utilize a simple but effective method to collect bursty topics in the query log data as follows:

• We choose bigrams as the basic presentation of bursty topics since bigrams present the information need of users more clearly and completely than unigrams and also have a larger coverage in the query log comparing to n-grams ($n \geq 3$).

• We only consider the bigram queries which appear more frequently than a threshold s per month. This is reasonable since a bursty event usually causes a large volume of search activities.

• Let $f_{max}(q)$ be the maximum search volume of a query q in one day (i.e., day d). Let $\hat{f}_{-5}(q)$ be the *upper* bound of the daily search volume of q outside a time window of 5 days centered at day d. If $f_{max}(q)$ is "significantly higher" than $\hat{f}_{-5}(q)$ (i.e., $r_m = \hat{f}_{max}(q)/f_{-5}(q) > m$), we consider q as a query with a spike pattern (m is an empirical threshold).

• The ratio above may be vulnerable to the query that has more than one spike. To solve this, we define $\bar{f}_{-5}(q)$ as the *average* of daily search volume of q outside the same time window. This gives us an alternative ratio $r_a = f_{max}(q)/\bar{f}_{-5}(q)$. We further balance these two ratios by ranking the bursty queries using

$$score(q) = \alpha \cdot r_m(q) + (1 - \alpha) \cdot r_a(q) \quad (1)$$

By setting $s = 20$, $m = 2.5$, $\alpha = 0.8$ (based on several tests), we select the top 130 bigram queries which form the pool of bursty topics for our analysis. Table 1 shows some of these topics, covering multiple domains: politics, science, art, sports, entertainment, etc.

ID	Topic	ID	Topic
1	kentucky election	66	orlando hernandez
2	indiana election	75	daniel biechele
8	van goph	81	hurricane forecast
24	north korea	92	93 memorial
34	pacific quake	113	holloway case
52	florida fires	128	stephen colbert
63	hunger strike	130	bear attack

Table 1: Examples of News Topics

3.2 Context extraction from multiple sources

Once we select the pool of bursty topics, we gather the contexts of each topic from multiple sources: query log, news media, blog media, and social bookmarks. We assume that the language in these contexts will reflect the reactions of the bursty events in corresponding online media.

3.2.1 Super query context

The most straightforward context of bursty events in web search is the query string, which directly reflects the users' interests and perspectives in the topic. We therefore define the first type of context of a bursty topic in query log as the set of surrounding terms of that bursty bigram in the (longer) queries. For example, the word *aftermath* in the query "haiti earthquake aftermath" is a term in the context of the bursty topic *haiti earthquake*.

Formally, we define a *Super Query* of a bursty topic t, $sq(t)$, as the query which contains the bigram query t lexically as a substring. For each bursty topic t, we scan the whole query log Q and retrieve all the super queries of t to form the context which is represented by $SQ(t)$.

$$SQ(t) = \{q | q \in Q \quad and \quad q = sq(t)\}$$

[1] Now known as Bing: www.bing.com

$SQ(t)$ is defined as the super query context of t. For example, the super query context of ”kentucky election” contains terms such as “2006,” “results,” “christian county,” etc. These terms indicate what aspects the users are most interested in Kentucky Election during May 2006.

The super query context is widely explored by search engines to provide query expansion and query completion (Jones et al., 2006).

3.2.2 Query session context

Another interesting context of a bursty topic in query log is the sequence of queries that a user searches after he submitted the bursty query q. This context usually reflects how a user reformulates the representation of his information need and implicitly clarifies his interests in the topic.

We define a *Query Session* containing a bursty topic t, $qs(t)$, as the queries which are issued by the same user after he issued t, within 30 minutes. For each bursty topic t, we collect all the $qs(t)$ to form the query session context of t, $QS(t)$:

$$QS(t) = \{q | q \in Q \quad and \quad q \in qs(t)\}$$

In web search, the query session context is usually utilized to provide query suggestion and query reformulation (Radlinski and Joachims, 2005).

3.2.3 News contexts

News articles written by critics and journalists reflect the reactions and perspectives of such professional group of people to a bursty event. We collect news articles about these 130 bursty topics from Google News[2], by finding the most relevant news articles which (1) match the bursty topic t, (2) were published in May, 2006, and (3) were published by any of the five major news medias: CNN, NBC, ABC, New York Times and Washington Post.

We then retrieve the *title* and *body* of each news article. This provides us two contexts of each bursty topic t: the set of relevant news titles, $NT(t)$, and the set of relevant news bodies, $NB(t)$.

3.2.4 Blog contexts

Compared with news articles, blog articles are written by common users in the online communities, which are supposed to reflect the reactions and opinions of the public to the bursty events. We collect blog articles about these 130 topics from Google Blog[3], by finding the most relevant blog articles which (1) match the bursty topic t, (2) were published in May, 2006 (3) were published in the most popular blog community, Blogspot[4]. We then retrieve the *title* and *body* of each relevant blog post respectively. This provides another two contexts: the set of relevant blog titles, $BT(t)$, and the set of relevant blog bodies, $BB(t)$.

3.2.5 Social bookmarking context

Social bookmarks form a new source of social media that allows the users to tag the webpages they are interested in and share their tags with others. The tags are supposed to reflect how the users describe the content of the pages and their perspectives of the content in a concise way.

We use a sample of Delicious[5] bookmarks in May, 2006, which contains around $1.37M$ unique URLs. We observe that the bursty bigram queries are also frequently used as tags in Delicious. We thus construct another context of bursty events by collecting all the tags that are used to tag the same URLs as the bursty topic.

Formally, we define $DT(t)$ as the context of social tags of a topic t,

$$DT(t) = \{tag | \exists url, \quad s.t. \quad tag, t \in B(url)\},$$

where url is a URL and $B(url)$ stands for the set of all bookmarks of url.

3.3 Context Statistics

Now we have constructed the set of 130 bursty topics and 7 corresponding contexts from various sources. We believe that these contexts well represent the various types of online media and sources.

For each context, we then clean the data by removing stopwords and the bursty topic keywords themselves. We then represent it as either the set of unigrams or bigrams from this context. Table 2 shows the basic statistics of each context:

From Table 2 we observe the following facts:

- The query session context covers more terms (both unigrams and bigrams) than the super query

[2] http://news.google.com/

[3] http://blogsearch.google.com/

[4] http://www.blogspot.com/

[5] http://delicious.com/

	N	T_S	M_S	A_U	M_U	A_B	M_B
SQ	130	76k	5.3k	32.7	390	24.3	235
QS	126	108k	5.8k	224	1.5k	150	1062
NT	118	4.7k	411	105	627	102	722
NB	118	4.7k	411	4.7k	22k	22k	257k
BT	128	5.8k	99	184	459	169	451
BB	128	5.8k	99	4.1k	15k	12k	69k
DT	71	2.3k	475	137	2.0k	N/A	N/A

N: The number of topics covered
T_S: The total number of records/documents
M_S: The max number of records/documents per topic
A_U: The avg number of unique unigrams per topic
M_U: The max number of unique unigrams
A_B: The avg number of unique bigrams
M_B: The max number of unique bigrams

Table 2: Basic statistics of collections

context. In both contexts, the average number of unique bigrams is smaller than unigrams. This is because queries in search are usually very short. After removing stopwords and topic keywords, quite a few queries have no bigram in these contexts.

• News articles and blog articles cover most of the bursty topics and contain a rich set of unigrams and bigrams in the corresponding contexts.

• The Delicious context only covers less than 60% of bursty topics. We couldn't extract bigrams from bookmarks since delicious provides a "bag-of-tags" interface.

In Section 4, we present a comprehensive analysis of these different contexts of bursty topics, with three different types of comparison.

4 Experiment

In this section, we present a comprehensive comparative analysis of the different contexts, which represent the reactions to the bursty topics in corresponding sources.

4.1 Similarity & Predictability analysis

Our first task is to compare the content similarity of these sources. This will help us to understand how well the language usage in one context can be leveraged to predict the language usage in another context. This is especially useful to predict the content in web search. By representing each context of a bursty topic as a vector space model of unigrams/bigrams, we first compute and compare the average *cosine similarity* between contexts. We only include contexts with more than 5 unigram/bigrams into this comparison. The results are shown in Table A and Table B, respectively. Each table is followed by a heat map to visualize the pattern.

To investigate how well one source can predict the content of another, we also represent each context of a bursty topic as a unigram/bigram language model and compute the *Cross Entropy* (Kullback and Leibler, 1951) between every pairs of contexts. Cross Entropy measures how certain one probability distribution predicts another. We calculate such measure based on the following definition:

$$H_{CE}(m||n) = H(m) + D_{KL}(m||n)$$

We smooth the unigram language models using Laplace smoothing (Field, 1988) and the bigram language models using Katz back-off model (Katz, 1987).

The results are shown in Table C and Table D, followed by the corresponding heat maps. For each value $H_{CE}(m||n)$ in the table cell, m stands for the context in the row and n stands for the context in the column. Please note that in Figure 3, 4, a larger H_{CE} value corresponds to a lighter cell.

4.1.1 Results

From the results shown in Table A-D, or in Figure 1- 4 more visually, some interesting phenomena can be observed:

• Compared with other contexts, query session is much more similar to the super query. This makes sense because many super queries would be included in the query session.

• Compared with news and blog, the delicious context is closer to the query log context. In fact, delicious is reasonably close to all the other contexts. This means social tags could be an effective source to enhance bursty topics in web search in terms of query suggestion. However, as Table 2 shows, only less than 60% of topics can be covered by delicious tag. We have to explore other sources to make a comprehensive prediction.

• In the news and blog contexts, the title contexts are more similar to the query contexts than the body contexts. This may be because titles usually concisely describe the topic while bodies contain much more details and irrelevant contents.

Context	SQ	QS	NT	NB	BT	BB	DT
SQ	1.0	0.405	0.122	0.072	0.119	0.061	0.188
QS	0.405	1.0	0.049	0.062	0.066	0.054	0.112
NT	0.122	0.049	1.0	0.257	0.186	0.152	0.120
NB	0.072	0.062	0.257	1.0	0.191	0.362	0.114
BT	0.119	0.066	0.186	0.191	1.0	0.242	0.141
BB	0.061	0.054	0.152	0.362	0.242	1.0	0.107
DT	0.188	0.112	0.120	0.114	0.141	0.107	1.0

Table A: Cosine similarity for unigram vectors

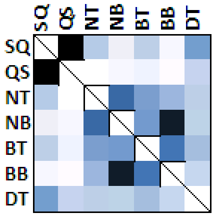


Figure 1: Heat map of table A

Source	SQ	QS	NT	NB	BT	BB
SQ	1.0	0.290	0.028	0.024	0.047	0.027
QS	0.290	1.0	0.004	0.010	0.011	0.009
NT	0.028	0.004	1.0	0.041	0.026	0.011
NB	0.024	0.010	0.041	1.0	0.023	0.040
BT	0.047	0.011	0.026	0.023	1.0	0.044
BB	0.027	0.009	0.011	0.040	0.044	1.0

We do not build bigram vector for DT

Table B: Cosine similarity for bigram vectors

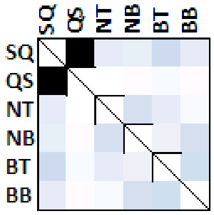


Figure 2: Heat map of table B

Source	SQ	QS	NT	NB	BT	BB	DT
SQ	1.698	4.911	7.538	8.901	7.948	9.050	7.498
QS	7.569	3.842	9.487	11.130	9.997	11.546	8.972
NT	8.957	10.868	3.718	7.946	9.006	9.605	8.825
NB	11.217	12.897	11.317	7.241	12.282	11.582	11.739
BT	9.277	11.084	9.085	10.295	4.637	9.365	9.180
BB	11.053	12.842	11.593	11.742	12.001	7.232	11.525
DT	8.457	9.794	8.521	9.511	8.831	9.473	2.990

Table C: Cross entropy for unigram distribution

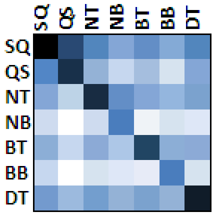


Figure 3: Heat map of table C

Source	SQ	QS	NT	NB	BT	BB
SQ	1.891	2.685	4.290	4.540	4.319	4.607
QS	6.800	3.430	8.144	9.049	8.528	9.304
NT	5.444	5.499	3.652	4.733	5.106	5.218
NB	11.572	11.797	11.254	8.731	11.544	11.073
BT	5.664	5.674	5.503	5.495	4.597	5.301
BB	10.745	10.796	10.517	10.455	10.526	8.518

We do not build bigram distribution for DT

Table D: Cross Entropy for bigram distribution

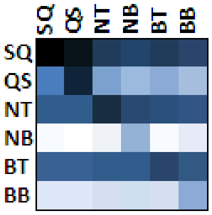


Figure 4: Heat map of table D

$H_{CE}(SQ||n)$

n:	NT	BT	NB	BB
Uni:	**7.538**	7.948	**8.901**	9.050
Bi:	**4.290**	4.319	**4.540**	4.607

$H_{CE}(m||SQ)$

m:	NT	BT	NB	BB
Uni:	**8.927**	9.277	11.217	**11.053**
Bi:	**5.445**	5.664	11.572	**10.745**

Table 3: Cross-entropy among three sources

• News would be a better predictor of the query than blog in general. This is interesting, which indicates that many search activities may be initialized by reading the news.

• News and blogs are much more similar to each other than query logs. We hypothesize that this result reflects the behavior how people write blogs about bursty events – typically they may have read several news articles before writing their own blog. In the blog, they may directly quote or retell a part of the news article and then add their opinion.

• Table 3 reveals the generation relations among three sources: query, news and blog. From the upper table, we can observe that queries are more likely to be generated by news articles, rather than blog articles. From the lower table, we can observe that queries are more likely to generate blog articles(body), rather than news articles(body). This result is quite interesting, which indicates the users' actual behaviors: when a bursty event happens, users would search them from web *after* they read it from some *news articles*. And users would write their own blogs to discuss the event *after* they retrieve and digest information from the web.

• From Table 3 we also find that queries are more likely to generate news *title*, rather than blog *title*. It is natural since blogs are written by kinds of people. The content especially the title part contains more uncertainty.

4.1.2 Case study

We then conduct the analysis to the level of individual topics. Table 4 shows the correlation of each pair of contexts, computed based on the similarity between topics in SQ and corresponding topics in these contexts. We can observe that $News$ and $Blog$ are correlated with each other tightly. If one is a good predictor of bursty queries, the other one also tends to be.

	QS	NT	NB	BT	BB	DT
QS		0.46	0.59	0.58	**0.75**	0.46
NT			**0.73**	**0.79**	0.59	0.61
NB				**0.71**	**0.68**	0.61
BT					**0.78**	0.59
BB						0.48

Table 4: Correlations of the similarity with SQ

For some topics like "stephen colbert," and "three gorges," both $News$ and $Blog$ are quite similar to the queries, which implies some intrinsic properties (*coherence*) of these topics: users would refer to the same content when using the topic terms in different sources.

We also find that a few topics like "hot dogs," "bear attack," for which the similarity of $(SQ, News)$ and $(SQ, Blog)$ are both low. It is probably because these topics are too diverse and carries a lot of *ambiguity*.

Although in most cases they are correlated, sometimes $News$ and $Blog$ show different trends in the similarity to the queries. For example, $News$ is quite similar to the queries on the topics such as "holloway case" and "jazz fest" while $Blog$ is dissimilar. For these *unfamiliar* topics, users possibly search the web "after" they read the news articles and express their diverse opinions in the blog. In contrast, on the topics like "insurance rate" or "consolidation loans," $Blog$ is similar to the queries while $News$ is not. For these *daily-life-related* queries, users would express the similar opinions when they search or write blogs, while news articles typically report such "professional" viewpoints.

4.2 Coverage analysis

Are social bookmarks the best source to predict bursty content in search? It looks so from the similarity comparison, *if they have a good coverage of search contents*. In this experiment, we analyze the coverage of query contexts in other contexts in a systematic way. If the majority of terms in the super query context would be covered by a small proportion of top words from another source, this source has the potential.

4.2.1 Unigram coverage

We first analyze the coverage of unigrams from the super query context in four other contexts: QS, DT, $News$ (the combination of NT and NB) and $Blog$ (the combination of BT and BB) to compare with SQ. For each source, we rank the unigrams by frequency. Figure 5(a) shows the average trend of SQ-unigram coverage in different sources. The x-coordinate refers to the ratio of top unigrams in one source to the number of unigrams in SQ. For example, if SQ contains n unigrams, the ratio 2 stands for the top $2n$ unigrams in the other source. The y-coordinate refers to the coverage rate of SQ. We can observe that:

• Query Session naturally covers most of the super query terms (over 70%).

• Though delicious tags are more similar to queries than news and blog, as well as a relatively higher coverage rate than the other two while size ratio is small, the overall coverage rate is quite low: only 21.28%. Note that this is contradict to existing comparative studies between social bookmarks and search logs (Bischoff et al., 2008). Clearly, when considering bursty queries, the coverage and effectiveness of social bookmarks is much lower than considering all queries. Handling bursty queries is much more difficult; only using social bookmarks to predict queries is not a good choice. Other useful sources should be enrolled.

• As the growth of the size ratio, the coverage rate of news and blogs are both gradually increased. When stable, both of them arrive at a relatively high level (news: 66.36%, blog: 63.80%), which means news and blogs have a higher potential to predict the bursty topics in search. Moreover, in most cases, news is still prior to blog – not only the overall rate, but also the size ratio comparison while the coverage rate reaches 50% (news:109 $<$ blog:183).

4.2.2 Bigram Coverage

Also we analyze the bigram coverage. This time we only have 3 sources (no DT). We rank the bigrams by the pointwise mutual information instead of frequency, since not all the bigrams are "real" collocations. Figure 5(b) shows the results.

Different from the unigram coverage, except that the query session can naturally keep a high coverage rate (66.07%), both news and blog cover poorly. For this issue, we should re-consider the behavior that users search and write articles. News or blog articles consist of completed sentences and paragraphs which would contain plenty of meaningful bigrams. However search queries consist of keywords – relatively discrete and regardless of order. Therefore, except some proper nouns such as person's name, a lot of bigrams in the query log are formed in an ad-hoc way. Since the different expressions of search and writing, detecting unigrams is more informational than bigrams.

4.3 Coherence analysis

The above two experiments discuss the inter-relations among different contexts. In this section we will discuss the inner-relation within each particular context – when it comes to a particular bursty topic, how coherent is the information in each context? Does the discussion keep consistent, or slip into ambiguity?

We represent all the terms forming each context of a bursty topic as a weighted graph: $G = (V, E, W)$, where each $v \in V$ stands for each term, w_v stands for the weight of vertex v in G, and each $e \in E$ stands for the semantic closeness between a pair of terms (u, v) measured by $sim(u, v)$. We define the *density* of such a semantic graph as follows:

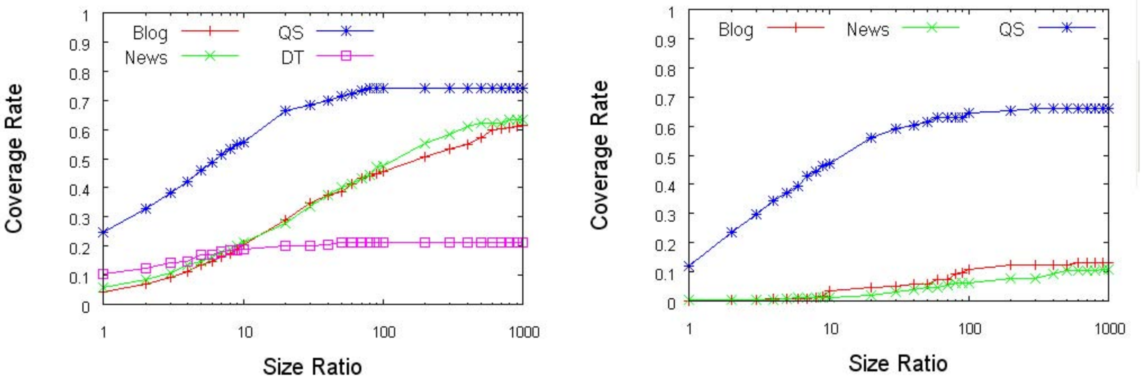
$$Den(G) = \frac{\Sigma_{u,v \in V, u \neq v} sim(u, v) w_u w_v}{\Sigma_{u,v \in V, u \neq v} w_u w_v} \quad (2)$$

If $sim(u, v)$ values the semantic similarity between u and v, a high value of $Den(G)$ implies that the whole context is semantically consistent. Otherwise, it may be diverse or ambiguous.

We build the graph of each context based on WordNet[6]. For a pair of words, WordNet provides a series of measures of the semantic similarity (Pedersen et al., 2004). We use the Path Distance Similarity (*path* for short) and Lin Similarity (*lin* for short) to measure $sim(u, v)$. Both measures range in $[0, 1]$.

For the convenience of computation, we choose the top 1100 unigrams ranked by term frequency in each source (if any) to represent the whole context on one specific topic.

[6] http://wordnet.princeton.edu/



(a) Unigram

(b) Bigram

Figure 5: Coverage results

4.3.1 Overall

Table 5 shows the average overall density of each sources over all the topics. From the table we can observe that QS has the lowest density in both of the measures. It is because the queries in one user session can easily shift to other (irrelevant) topics even in a short time.

Source	*path*	*lin*
SQ	0.098	0.128
QS	0.071	0.082
NT	0.103	0.129
NB	0.109	0.139
BT	0.099	0.109
BB	0.116	0.147
DT	0.102	0.127

Table 5: Overall Density

Another interesting phenomenon comes out that for either news or blog, the *body* is denser than the *title*, even if the body context contains much more terms. It can be explained by the roles of the title and the body in one article: the title contains a series of words which briefly summarize a topic while the body part would describe and discuss the title in details. When it maps to the semantic word network, the title tends to contain the vertices scattered in the graph, while the context of the body part would add more semantically related vertices around the original vertices to strength the relations. Thus, the body part has a higher density than the title part.

4.3.2 The trend analysis

Figure 6 shows the tendency of the density in each source. The x-coordinate refers to the TopN unigrams ranked by the term frequency in each source. From Figure 6 we can find that in most cases, the average density will gradually decrease while less important terms are added, which implies that the most important terms are denser, and other terms would disperse the topic.

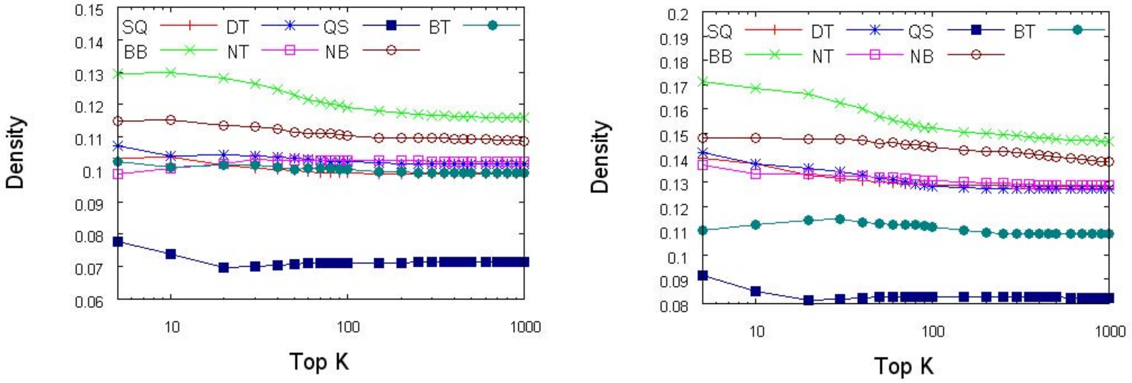
To better evaluate this tendency of each source, Table 6 shows the change rate of the highest density to the overall density measured by *lin*. We can easily find the following facts:

• The highest density is achieved when a small proportion of top terms are counted (6 sources for Top5 and one for Top20), which also supports our hypothesis: the more important, the more coherent.

• BB's density drops the fastest of all (15.1%), following by DT(10.6%). It may be because both blog and delicious tag are generated by many users. And the diversity of the users leads to different prospectives, which dilutes the context significantly.

• Both NT and NB drop quite slowly (5.8%, 6.8%), which means the professional journalists would have the relatively similar prospectives on the same topic. Thus the topic does not disperse too much. BT also keeps a high stability.

• Compared with news, blog is easier to disperse, which can be reflected by the density comparison between NT and BT. Although the density of BB is still higher than NT, we should notice that these two sources are not completely covered – about $3/4$ unigrams in these two contexts are not included in



(a) *path*

(b) *lin*

Figure 6: Trend of Density

the semantic networks. The curves clearly shows BB dropped faster than NB. One can expect that NB becomes denser than BB if all the unigrams in both sources are included in the network.

Source	Highest Den.	Overall Den.	Change
SQ	0.140(Top5)	0.128	-8.6%
QS	0.091(Top5)	0.082	-9.9%
NT	0.137(Top5)	0.129	-5.8%
NB	0.148(Top5)	0.139	-6.8%
BT	0.114(Top20)	0.109	-4.4%
BB	0.172(Top5)	0.147	-15.1%
DT	0.142(Top5)	0.127	-10.6%

Table 6: Tendency analysis of Density (Lin)

4.3.3 Case Study

From these 130 news topics, some of them shows a special tendency of coherence. For example, when more words are included, the density of the topic“three gorge” drops rapidly in most of the sources. The topic “florida fires” has the same trend. These topics are typically “*focus*” topics, which means users clearly pursue the unique event while they use these terms. Thus, the density in top unigrams is very high. It drops rapidly since users' personal interests and opinions toward to this event will be enrolled gradually.

In contrast, some topics like “heather mills,”, “insurance rate” express differently: their densities gradually increase with the growth of the terms. By observing these topics we find they are usually diverse topics (e.g: famous person name or entity name), which may lead to diverse search intentions of users. So the density of top unigrams is low and gradually increased since one main aspect is probably strengthened.

5 Conclusion and Future work

In this paper, we have studied and compared how the web content reacts to bursty events in multiple contexts of web search and online media. After a series of comprehensive experiments including content similarity and predictability, the coverage of search content, and semantic diversity, we found that social bookmarks are not enough to predict the queries because of a low coverage. Other sources like news and blogs need to be added. Furthermore, news can be seen as a consistent source which would not only trigger the discussion of bursty events in blogs but also in search queries.

When the target is to diversify the search results and query suggestions, blogs and social bookmarks are potentially useful accessory sources because of the high diversity of content.

Our work serves as a feasibility investigation of query suggestion for bursty events. Future work would address on how to systematically predict and recommend the bursty queries using online media, as well as a reasonable evaluation metrics upon it.

Acknowledgments

We thank Prof. Kevin Chang for his support in data and useful discussion. We thank the three anonymous reviewers for their useful comments. This work is in part supported by the National Science Foundation under award number IIS-0968489.

References

Jon Kleinberg 2003. Bursty and Hierarchical Structure in Streams *Data Mining and Knowledge Discovery*, Vol 7(4):373-397

Daniel Gruhl, R. Guha, Ravi Kumar, Jasmine Novak and Andrew Tomkins 2005. The Predictive Power of Online Chatter *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, 78-87.

Ravi Kumar, Jasmine Novak, Prabhakar Raghavan and Andrew Tomkins 2003. On the Bursty Evolution of Blogspace *WWW '03: Proc. of the 12th International World Wide Web Conference*, 568-576.

Michail Vlachos and Christopher Meek and Zografoula Vagena and Dimitrios Gunopulos 2004. Identifying similarities, periodicities and bursts for online search queries *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, 131-142.

Evgeniy Gabrilovich, Susan Dumais and Eric Horvitz 2004. Newsjunkie: providing personalized newsfeeds via analysis of information novelty *WWW '04: Proceedings of the 13th international conference on World Wide Web*, 482-490.

Eytan Adar, Daniel S. Weld, and Brian N. Bershad and Steven S. Gribble 2007. Why we search: visualizing and predicting user behavior *WWW '07: Proceedings of the 16th international conference on World Wide Web*, 161-170.

Aixin Sun, Meishan Hu and Ee-Peng Lim 2008. Searching blogs and news: a study on popular queries *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 729-730.

JeanPhilippe Cointet, Emmanuel Faure and Camille Roth 2008. Intertemporal topic correlations in online media : A Comparative Study on Weblogs and News Websites *ICWSM '08: International Coference on Weblogs and Social Media*

Levon Lloyd, Prachi Kaulgud and Steven Skiena 2006. Newspapers vs. Blogs: Who Gets the Scoop? *AAAI Spring Symposium on Computational Approaches to Analyzing Weblogs*

Michael Gamon, Sumit Basu, Dmitriy Belenko, Danyel Fisher, Matthew Hurst, and Arnd Christian Konig 2008. BLEWS: Using Blogs to Provide Context for News Articles *ICWSM '08: International Coference on Weblogs and Social Media*

Sanjay Sood, Sara Owsley, Kristian Hammond and Larry Birnbaum 2007. TagAssist: Automatic Tag Suggestion for Blog Posts *ICWSM '07: International Coference on Weblogs and Social Media*

Fernando Diaz 2009. Integration of news content into web results *WSDM '09: Proceedings of the Second ACM International Conference on Web Search and Data Mining*, 182-191.

Beate Krause, Andreas Hotho and Gerd Stumme 2008. A Comparison of Social Bookmarking with Traditional Search *Advances in Information Retrieval*, Vol 4956/2008:101-113.

Paul Heymann, Georgia Koutrika and Hector Garcia-Molina 2008. Can social bookmarking improve web search? *WSDM '08: Proceedings of the international conference on Web search and web data mining*, 195-206.

Shenghua Bao, Guirong Xue, Xiaoyuan Wu, Yong Yu, Ben Fei and Zhong Su 2007. Optimizing web search using social annotations *WWW '07: Proceedings of the 16th international conference on World Wide Web*, 501-510.

Kerstin Bischoff, Claudiu S. Firan, Wolfgang Nejdl and Raluca Paiu 2008. Can all tags be used for search? *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, 193-202.

Rosie Jones, Benjamin Rey and Omid Madani 2006. Generating query substitutions *Proceedings of the 15th international conference on World Wide Web,*, 387-396.

Filip Radlinski and Thorsten Joachims 2005. Query chains: learning to rank from implicit feedback *Proceedings of the 11th ACM SIGKDD international conference on Knowledge discovery in data mining*, 239-248.

Qiaozhu Mei, Dengyong Zhou and Kenneth Church 2008. Query suggestion using hitting time *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, 469-478

Andrei Broder 2002. A Taxonomy of Web Search *SIGIR Forum*, Vol 36(2):3-10.

Solomon Kullback and Richard Leibler 1951. On Information and Sufficience *Annals of Mathematical Statistics*, Vol 22(1):79-86.

David A. Field 1988. Laplacian Smoothing and Delaunay Triangulations *Communications in Applied Numerical Methods*, Vol 4:709-712.

Stephen M. Katz 1987 Estimation of probabilities from sparse data for the language model component of a speech recogniser *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3), 400-401.

Ted Pedersen, Siddharth Patwardhan and Jason Michelizzi 2004. WordNet::Similarity: measuring the relatedness of concepts *PHLT-NAACL '04: Demonstration Papers at HLT-NAACL 2004 on XX*, 38-41.

Learning First-Order Horn Clauses from Web Text

Stefan Schoenmackers, Oren Etzioni, Daniel S. Weld
Turing Center
University of Washington
Computer Science and Engineering
Box 352350
Seattle, WA 98125, USA
stef,etzioni,weld@cs.washington.edu

Jesse Davis
Katholieke Universiteit Leuven
Department of Computer Science
POBox 02402 Celestijnenlaan 200a
B-3001 Heverlee, Belgium
jesse.davis@cs.kuleuven.be

Abstract

Even the entire Web corpus does not *explicitly* answer all questions, yet inference can uncover many *implicit* answers. But where do inference rules come from?

This paper investigates the problem of *learning* inference rules from Web text in an unsupervised, domain-independent manner. The SHERLOCK system, described herein, is a first-order learner that acquires over 30,000 Horn clauses from Web text. SHERLOCK embodies several innovations, including a novel rule scoring function based on Statistical Relevance (Salmon et al., 1971) which is effective on ambiguous, noisy and incomplete Web extractions. Our experiments show that inference over the learned rules discovers three times as many facts (at precision 0.8) as the TEXTRUNNER system which merely extracts facts explicitly stated in Web text.

1 Introduction

Today's Web search engines locate pages that match keyword queries. Even sophisticated Web-based Q/A systems merely locate pages that contain an *explicit* answer to a question. These systems are helpless if the answer has to be inferred from multiple sentences, possibly on different pages. To solve this problem, Schoenmackers *et al.*(2008) introduced the HOLMES system, which infers answers from tuples extracted from text.

HOLMES's distinction is that it is domain independent and that its inference time is *linear* in the size of its input corpus, which enables it to scale to the Web. However, HOLMES's Achilles heel is that it requires hand-coded, first-order, Horn clauses as input. Thus, while HOLMES's inference *run time* is highly scalable, it requires substantial labor and expertise to hand-craft the appropriate set of Horn clauses for each new domain.

Is it possible to *learn* effective first-order Horn clauses automatically from Web text in a domain-independent and scalable manner? We refer to the set of ground facts derived from Web text as *open-domain theories*. Learning Horn clauses has been studied extensively in the Inductive Logic Programming (ILP) literature (Quinlan, 1990; Muggleton, 1995). However, learning Horn clauses from open-domain theories is particularly challenging for several reasons. First, the theories denote instances of an unbounded and unknown set of relations. Second, the ground facts in the theories are noisy, and incomplete. Negative examples are mostly absent, and certainly we cannot make the closed-world assumption typically made by ILP systems. Finally, the names used to denote both entities and relations are rife with both synonyms and polysymes making their referents ambiguous and resulting in a particularly noisy and ambiguous set of ground facts.

This paper presents a new ILP method, which is optimized to operate on open-domain theories derived from massive and diverse corpora such as the Web, and experimentally confirms both its effectiveness and superiority over traditional ILP algorithms in this context. Table 1 shows some example rules that were learned by SHERLOCK.

This work makes the following contributions:

1. We describe the design and implementation of the SHERLOCK system, which utilizes a novel, unsupervised ILP method to learn first-order Horn clauses from open-domain Web text.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1088–1098,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

IsHeadquarteredIn(Company, State) :-
IsBasedIn(Company, City) ∧ IsLocatedIn(City, State);

Contains(Food, Chemical) :-
IsMadeFrom(Food, Ingredient) ∧ Contains(Ingredient, Chemical);

Reduce(Medication, Factor) :-
KnownGenericallyAs(Medication, Drug) ∧ Reduce(Drug, Factor);

ReturnTo(Writer, Place) :- BornIn(Writer, City) ∧ CapitalOf(City, Place);

Make(Company1, Device) :- Buy(Company1, Company2) ∧ Make(Company2, Device);

Table 1: Example rules learned by SHERLOCK from Web extractions. Note that the italicized rules are unsound.

2. We derive an innovative scoring function that is particularly well-suited to unsupervised learning from noisy text. For Web text, the scoring function yields more accurate rules than several functions from the ILP literature.
3. We demonstrate the utility of SHERLOCK's automatically learned inference rules. Inference using SHERLOCK's learned rules identifies three times as many high quality facts (*e.g.*, precision $\geq$ 0.8) as were originally extracted from the Web text corpus.

The remainder of this paper is organized as follows. We start by describing previous work. Section 3 introduces the SHERLOCK rule learning system, with Section 3.4 describing how it estimates rule quality. We empirically evaluate SHERLOCK in Section 4, and conclude.

2 Previous Work

SHERLOCK is one of the first systems to learn first-order Horn clauses from open-domain Web extractions. The learning method in SHERLOCK belongs to the Inductive logic programming (ILP) subfield of machine learning (Lavrac and Dzeroski, 2001). However, classical ILP systems (*e.g.*, FOIL (Quinlan, 1990) and Progol (Muggleton, 1995)) make strong assumptions that are inappropriate for open domains. First, ILP systems assume high-quality, hand-labeled training examples for each relation of interest. Second, ILP systems assume that constants uniquely denote individuals; however, in Web text strings such as "dad" or "John Smith" are highly ambiguous. Third, ILP system typically assume complete, largely noise-free data whereas tuples extracted from Web text are both noisy and radically incomplete. Finally, ILP systems typically utilize negative examples, which are not available when learning from open-domain facts. One system that does not require negative examples is LIME (McCreath and Sharma, 1997); We compare SHERLOCK with LIME's methods in Section 4.3. Most prior ILP and Markov logic structure learning systems (*e.g.*, (Kok and Domingos, 2005)) are not designed to handle the noise and incompleteness of open-domain, extracted facts.

NELL (Carlson et al., 2010) performs coupled semi-supervised learning to extract a large knowledge base of instances, relations, and inference rules, bootstrapping from a few seed examples of each class and relation of interest and a few constraints among them. In contrast, SHERLOCK focuses mainly on learning inference rules, but does so without any manually specified seeds or constraints.

Craven *et al.*(1998) also used ILP to help information extraction on the Web, but required training examples and focused on a single domain.

Two other notable systems that learn inference rules from text are DIRT (Lin and Pantel, 2001) and RESOLVER (Yates and Etzioni, 2007). However, both DIRT and RESOLVER learn only a limited set of rules capturing synonyms, paraphrases, and simple entailments, not more expressive multi-part Horn clauses. For example, these systems may learn the rule X acquired $Y \implies X$ bought Y, which captures different ways of describing a purchase. Applications of these rules often depend on context (*e.g.*, if a person acquires a skill, that does not mean they bought the skill). To add the necessary context, ISP (Pantel et al., 2007) learned selectional preferences (Resnik, 1997) for DIRT's rules. The selectional preferences act as type restrictions

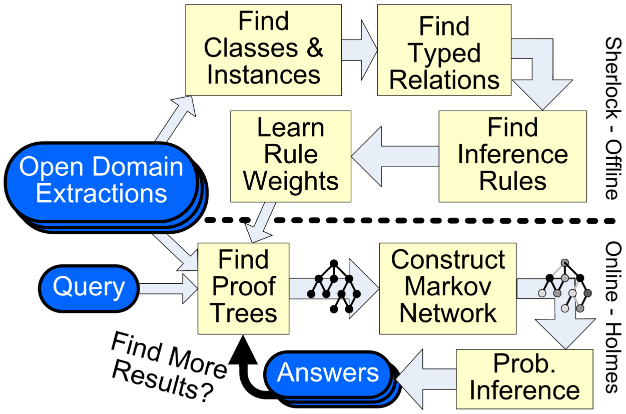


Figure 1: Architecture of SHERLOCK. SHERLOCK learns inference rules offline and provides them to the HOLMES inference engine, which uses the rules to answer queries.

on the arguments, and attempt to filter out incorrect inferences. While these approaches are useful, they are strictly more limited than the rules learned by SHERLOCK.

The Recognizing Textual Entailment (RTE) task (Dagan et al., 2005) is to determine whether one sentence entails another. Approaches to RTE include those of Tatu and Moldovan (2007), which generates inference rules from WordNet lexical chains and a set of axiom templates, and Pennacchiotti and Zanzotto (2007), which learns inference rules based on similarity across entailment pairs. In contrast with this work, RTE systems reason over full sentences, but benefit by being given the sentences and training data. SHERLOCK operates over simpler Web extractions, but is not given guidance about which facts may interact.

3 System Description

SHERLOCK takes as input a large set of open domain facts, and returns a set of weighted Horn-clause inference rules. Other systems (*e.g.*, HOLMES) use the rules to answer questions, infer additional facts, *etc.*

SHERLOCK's basic architecture is depicted in Figure 1. To learn inference rules, SHERLOCK performs the following steps:

1. Identify a "productive" set of classes and instances of those classes
2. Discover relations between classes
3. Learn inference rules using the discovered relations and determine the confidence in each rule

The first two steps help deal with the synonyms, homonyms, and noise present in open-domain theories by identifying a smaller, cleaner, and more cohesive set of facts to learn rules over.

SHERLOCK learns inference rules from a collection of open-domain extractions produced by TEXTRUNNER (Banko et al., 2007). The rules learned by SHERLOCK are input to an inference engine and used to find answers to a user's query. In this paper, SHERLOCK utilizes HOLMES as its inference engine when answering queries, and uses extracted facts of the form R(arg1, arg2) provided by the authors of TEXTRUNNER, but the techniques presented are more broadly applicable.

3.1 Finding Classes and Instances

SHERLOCK first searches for a set of well-defined classes and class instances. Instances of the same class tend to behave similarly, so identifying a good set of instances will make it easier to discover the general properties of the entire class.

Options for identifying interesting classes include manually created methods (WordNet (Miller et al., 1990)), textual patterns (Hearst, 1992), automated clustering (Lin and Pantel, 2002), and combinations (Snow et al., 2006). We use Hearst patterns because they are simple, capture how classes and instances are mentioned in Web text, and yield intuitive, explainable groups.

Hearst (1992) identified a set of textual patterns which indicate hyponymy (*e.g.*, '*Class* such as *Instance*'). Using these patterns, we extracted 29 million (instance, class) pairs from a large Web crawl. We then cleaned them using word stemming, normalization, and by dropping modifiers.

Unfortunately, the patterns make systematic errors (*e.g.*, extracting Canada as the name of a city from the phrase 'Toronto, Canada and other cities.') To address this issue, we discard the low frequency classes of each instance. This heuristic reduces the noise due to systematic error while still capturing the important senses of each word. Additionally, we use the extraction frequency to estimate the probability that a particular mention of an instance refers to each of its potential classes (*e.g.*, New York appears as a city 40% of the time, a state 35% of the time, and a place, area, or center the rest of the time).

Ambiguity presents a significant obstacle when learning inference rules. For example, the corpus contains the sentences ‘broccoli contains this vitamin’ and ‘this vitamin prevents scurvy,’ but it is unclear if the sentences refer to the same vitamin. The two main sources of ambiguity we observed are references to a more general class instead of a specific instance (*e.g.*, ‘vitamin’), and references to a person by only their first or last name. We eliminate the first by removing terms that frequently appear as the class name with other instances, and the second by removing common first and last names.

The 250 most frequently mentioned class names include a large number of interesting classes (*e.g.*, companies, cities, foods, nutrients, locations) as well as ambiguous concepts (*e.g.*, ideas, things). We focus on the less ambiguous classes by eliminating any class not appearing as a descendant of physical entity, social group, physical condition, or event in WordNet. Beyond this filtering we make no use of a type hierarchy and treat classes independently.

In our corpus, we identify 1.1 million distinct, cleaned (instance, class) pairs for 156 classes.

3.2 Discovering Relations between Classes

Next, SHERLOCK discovers how classes relate to and interact with each other. Prior work in relation discovery (Shinyama and Sekine, 2006) has investigated the problem of finding relationships between different classes. However, the goal of this work is to learn rules on top of the discovered typed relations. We use a few simple heuristics to automatically identify interesting relations.

For every pair of classes (C_1, C_2), we find a set of typed, candidate relations from the 100 most frequent relations in the corpus where the first argument is an instance of C_1 and the second argument is an instance of C_2. For extraction terms with multiple senses (*e.g.*, New York), we split their weight based on how frequently they appear with each class in the Hearst patterns.

However, many discovered relations are rare and meaningless, arising from either an extraction error or word-sense ambiguity. For example, the extraction ‘Apple is based in Cupertino’ gives some evidence that a fruit may possibly be based in a city. We attempt to filter out incorrectly-typed relations using two heuristics. We first discard any relation whose weighted frequency falls below a threshold, since rare relations are more likely to arise due to extraction errors or word-sense ambiguity. We also remove relations whose pointwise mutual information (PMI) is below a threshold T=$exp(2) \approx 7.4$:

$$PMI(R(C_1, C_2)) = \frac{p(R, C_1, C_2)}{p(R, \cdot, \cdot) * p(\cdot, C_1, \cdot) * p(\cdot, \cdot, C_2)}$$

where $p(R, \cdot, \cdot)$ is the probability a random extraction has relation R, $p(\cdot, C_1, \cdot)$ is the probability a random extraction has an instance of C_1 as its first argument, $p(\cdot, \cdot, C_2)$ is similar for the second argument, and $p(R, C_1, C_2)$ is the probability that a random extraction has relation R and instances of C_1 and C_2 as its first and second arguments, respectively. A low PMI indicates the relation occurred by random chance, which is typically due to ambiguous terms or extraction errors.

Finally, we use two TEXTRUNNER specific cleaning heuristics: we ignore a small set of stop-relations (‘be’, ‘have’, and ‘be *preposition*’) and extractions whose arguments are more than four tokens apart. This process identifies 10,000 typed relations.

3.3 Learning Inference Rules

SHERLOCK attempts to learn inference rules for each typed relation in turn. SHERLOCK receives a target relation, R, a set of observed examples of the relation, E^+, a maximum clause length k, a minimum support, s, and an acceptance threshold, t, as input. SHERLOCK generates all first-order, definite clauses up to length k, where R appears as the head of the clause. It retains each clause that:

1. Contains no unbound variables
2. Infers at least s examples from E^+
3. Scores at least t according to the score function

We now propose a novel score function, and empirically validate our choice in Sections 4.3 and 4.4.

3.4 Evaluating Rules by Statistical Relevance

The problem of evaluating candidate rules has been studied by many researchers, but typically in either a supervised or propositional context whereas we are learning first-order Horn-clauses from a noisy set of positive examples. Moreover, due to the incomplete nature of the input corpus and the imperfect yield of extraction—many true facts are not stated explicitly in the set of ground assertions used by the learner to evaluate rules.

The absence of negative examples, coupled with noise, means that standard ILP evaluation functions (*e.g.*, (Quinlan, 1990) and (Dzeroski and Bratko, 1992)) are not appropriate. Furthermore, when evaluating a particular rule with consequent C and antecedent A, it is natural to consider $p(C|A)$ but, due to missing data, this absolute probability estimate is often misleading: in many cases C will hold given A but the fact C is not mentioned in the corpus.

Thus to evaluate rules over extractions, we need to consider *relative* probability estimates. *I.e.*, is $p(C|A) \gg p(C)$? If so, then A is said to be *statistically relevant* to C (Salmon et al., 1971).

Statistical relevance tries to infer the simplest set of factors which explain an observation. It can be viewed as searching for the simplest propositional Horn-clause which increases the likelihood of a goal proposition g. The two key ideas in determining statistical relevance are discovering factors which substantially increase the likelihood of g (even if the probabilities are small in an absolute sense), and dismissing irrelevant factors.

To illustrate these concepts, consider the following example. Suppose our goal is to predict if New York City will have a storm (S). On an arbitrary day, the probability of having a storm is fairly low ($p(S) \ll 1$). However, if we know that the atmospheric pressure on that day is low, this substantially increases the probability of having a storm (although that absolute probability may still be small). According to the principle of statistical relevance, low atmospheric pressure (LP) is a factor which predicts storms (S :- LP), since $p(S|LP) \gg p(S)$.

The principle of statistical relevance also identifies and removes irrelevant factors. For example, let M denote the gender of New York's mayor. Since $p(S|LP, M) \gg p(S)$, it naïvely appears that storms in New York depend on the gender of the mayor in addition to the air pressure. The statistical relevance principle sidesteps this trap by removing any factors which are conditionally independent of the goal, given the remaining factors. For example, we observe $p(S|LP)$=$p(S|LP, M)$, and so we say that M is not statistically relevant to S. This test applies Occam's razor by searching for the simplest rule which explains the goal.

Statistical relevance appears useful in the open-domain context, since all the necessary probabilities can be estimated from only positive examples. Furthermore, approximating relative probabilities in the presence of missing data is much more reliable than determining absolute probabilities.

Unfortunately, Salmon defined statistical relevance in a propositional context. One technical contribution of our work is to lift statistical relevance to first order Horn-clauses as follows. For the Horn-clause $\mathtt{Head}(\mathtt{v}_1, ..., \mathtt{v}_n)$:-$\mathtt{Body}(\mathtt{v}_1, ..., \mathtt{v}_m)$ (where $\mathtt{Body}(\mathtt{v}_1, ..., \mathtt{v}_m)$ is a conjunction of function-free, non-negated, first-order relations, and $v_i \in V$ is the set of typed variables used in the rule), we say the body helps explain the head if:

1. Observing an instance of the body substantially increases the probability of observing the head.
2. The body contains no irrelevant (conditionally independent) terms.

We evaluate conditional independence of terms using ILP's technique of Θ-subsumption, ensuring there is no more general clause that is similarly predictive of the head. Formally, clause C_1 Θ-subsumes clause C_2 if and only if there exists a substitution Θ such that $C_1\Theta \subseteq C_2$ where each clause is treated as the set of its literals. For example, $\mathtt{R}(\mathtt{x}, \mathtt{y})$ Θ-subsumes $\mathtt{R}(\mathtt{x}, \mathtt{x})$, since $\{\mathtt{R}(\mathtt{x}, \mathtt{y})\}\Theta \subseteq \{\mathtt{R}(\mathtt{x}, \mathtt{x})\}$ when Θ=$\{\mathtt{y}/\mathtt{x}\}$. Intuitively, if C_1 Θ-subsumes C_2, it means that C_1 is more general than C_2.

Definition 1 *A first-order Horn-clause* $\mathtt{Head}(...)$:-$\mathtt{Body}(...)$ *is* statistically relevant *if* $p(\mathtt{Head}(...)|\mathtt{Body}(...)) \gg p(\mathtt{Head}(...))$ *and if there is no clause body* $\mathtt{B}'(...)\Theta \subseteq \mathtt{Body}(...)$ *such that* $p(\mathtt{Head}(...)|\mathtt{Body}(...)) \approx p(\mathtt{Head}(...)|\mathtt{B}'(...))$

In practice it is difficult to determine the probabilities exactly, so when checking for statistical relevance we ensure that the probability of the rule is at least a factor t greater than the probability of any subsuming rule, that is, $p(\mathtt{Head}(...)|\mathtt{Body}(...)) \geq t * p(\mathtt{Head}(...)|\mathtt{B}'(...))$

We estimate $p(\mathtt{Head}(...)|\mathtt{B}(...))$ from the observed facts by assuming values of $\mathtt{Head}(...)$ are generated by sampling values of $\mathtt{B}(...)$ as follows: for variables v_s shared between $\mathtt{Head}(...)$ and $\mathtt{B}(...)$, we sample values of v_s uniformly from all observed groundings of $\texttt{B}(...)$. For variables v_i, if any, that appear in $\texttt{Head}(...)$ but not in $\texttt{B}(...)$, we sample their values according to a distribution $p(v_i|class_i)$. We estimate $p(v_i|class_i)$ based on the relative frequency that v_i was extracted using a Hearst pattern with $class_i$.

Finally, we ensure the differences are statistically significant using the likelihood ratio statistic:

$$2N_r \sum_{\substack{\texttt{H}(...)\in \\ \{\texttt{Head}(...),\neg\texttt{Head}(...)\}}} p(\texttt{H}(...)|\texttt{Body}(...)) * log\frac{p(\texttt{H}(...)|\texttt{Body}(...))}{p(\texttt{H}(...)|\texttt{B}'(...))}$$

where $p(\neg\texttt{Head}(...)|\texttt{B}(...)) = 1-p(\texttt{Head}(...)|\texttt{B}(...))$ and N_r is the number of results inferred by the rule $\texttt{Head}(...)$:-$\texttt{Body}(...)$. This test is distributed approximately as χ^2 with one degree of freedom. It is similar to the *statistical significance* test used in mFOIL (Dzeroski and Bratko, 1992), but has two modifications since SHERLOCK doesn't have training data. In lieu of positive and negative examples, we use whether or not the inferred head value was observed, and compare against the distribution of a subsuming clause $\texttt{B}'(...)$ rather than a known prior.

This method of evaluating rules has two important differences from ILP under a closed world assumption. First, our probability estimates consider the fact that examples provide varying amounts of information. Second, statistical relevance finds rules with large increases in relative probability, not necessarily a large absolute probability. This is crucial in an open domain setting where most facts are false, which means the trivial rule that everything is false will have high accuracy. Even for true rules, the observed estimates $p(\texttt{Head}(...)|\texttt{Body}(...)) \ll 1$ due to missing data and noise.

3.5 Making Inferences

In order to benefit from learned rules, we need an inference engine; with its linear-time scalability, HOLMES is a natural choice (Schoenmackers et al., 2008). As input HOLMES requires a target atom $\texttt{H}(...)$, an evidence set E and weighted rule set R as input. It performs a form of *knowledge based model construction* (Wellman et al., 1992), first finding facts using logical inference, then estimating the confidence of each using a Markov Logic Network (Richardson and Domingos, 2006).

Prior to running inference, it is necessary to assign a weight to each rule learned by SHERLOCK. Since the rules and inferences are based on a set of noisy and incomplete extractions, the algorithms used for both weight learning and inference should capture the following characteristics of our problem:

C1. Any arbitrary unknown fact is highly unlikely to be true.

C2. The more frequently a fact is extracted from the Web, the more likely it is to be true. However, facts in E should have a confidence bounded by a threshold $p_{max} < 1$. E contains systematic extraction errors, so we want uncertainty in even the most frequently extracted facts.

C3. An inference that combines uncertain facts should be less likely than each fact it uses.

Next, we describe the needed modifications to the weight learning and inference algorithm to achieve the desired behavior.

3.5.1 Weight Learning

We use the discriminative weight learning procedure described by Huynh and Mooney (2008). Setting the weights involves counting the number of true groundings for each rule in the data (Richardson and Domingos, 2006). However, the noisy nature of Web extractions will make this an overestimate. Consequently, we compute $n_i(E)$, the number of true groundings of rule i, as follows:

$$n_i(E) = \sum_j \max_k \prod_{B(...)\in Body_{ijk}} p(B(...)) \qquad (1)$$

where E is the evidence, j ranges over heads of the rule, $Body_{ijk}$ is the body of the kth grounding for jth head of rule i, and $p(B(...))$ is approximated using a logistic function of the number of times $B(...)$ was extracted,[1] scaled to be in the range [0,0.75]. This models **C2** by giving increasing but bounded confidence for more frequently extracted facts. In practice, this also helps address **C3** by giving longer rules smaller values of n_i, which reflects that inferences arrived at through a combination of multiple, noisy facts should have lower confidence. Longer rules are also more likely to have multiple groundings that infer a particular head, so keeping only the most likely grounding prevents a head from receiving undue weight from any single rule.

[1] We note that this approximation is equivalent to an MLN which uses only the two rules defined in 3.5.2

Finally, we place a very strong Gaussian prior (*i.e.*, L_2 penalty) on the weights. Longer rules have a higher prior to capture the notion that they are more likely to make incorrect inferences. Without a high prior, each rule would receive an unduly high weight as we have no negative examples.

3.5.2 Probabilistic Inference

After learning the weights, we add the following two rules to our rule set:

1. `H`(...) with negative weight w_{prior}
2. `H`(...):-`ExtractedFrom`(`H`(...),$sentence_i$) with weight 1

The first rule models **C1**, by saying that most facts are false. The second rule models **C2**, by stating the probability of fact depends on the number of times it was extracted. The weights of these rules are fixed. We do not include these rules during weight learning as doing so swamps the effects of the other inference rules (*i.e.*, forces them to zero).

HOLMES attempts to infer the truth value of each ground atom `H`(...) in turn by treating all other extractions E in our corpus as evidence. Inference also requires computing $n_i(E)$ which we do according to Equation 1 as in weight learning.

4 Experiments

One can attempt to evaluate a rule learner by estimating the quality of learned rules, or by measuring their impact on a system that uses the learned rules. Since the notion of 'rule quality' is vague except in the context of an application, we evaluate SHERLOCK in the context of the HOLMES inference-based question answering system.

Our evaluation focuses on three main questions:

1. Does inference utilizing learned Horn rules improve the precision/recall of question answering and by how much?
2. How do different rule-scoring functions affect the performance of learning?
3. What role does each of SHERLOCK's components have in the resulting performance?

4.1 Methodology

Our objective with rule learning was to improve the system's ability to answer questions such as 'What foods prevent disease?' So we focus our evaluation on the task of computing as many instances as possible of an atomic pattern `Rel`(x, y). In this example, `Rel` would be bound to 'Prevents', x would have type 'Food' and y would have type 'Disease.'

But which relations should be used in the test? There is a large variance in behavior across relations, so examining any particular relation may give misleading results. Instead, we examine the global performance of the system by querying HOLMES for all open-domain relations identified in Section 3.2 as follows:

1. Score all candidate rules according to the rule scoring metric M, accept all rules with a score at least t_M (tuned on a small development set of rules), and learn weights for all accepted rules.
2. Find all facts inferred by the rules and use the rule weights to estimate the fact probabilities.
3. Reduce type information. For each fact, (*e.g.*, `BasedIn`(`Diebold`, `Ohio`)) which has been deduced with multiple type signatures (*e.g.*, Ohio is both a state and a geographic location), keep only the one with maximum probability (*i.e.*, conservatively assuming dependence).
4. Place all results into bins based on their probabilities, and estimate the precision and the number of correct facts in the bin using a random sample.

In these experiments we consider rules with up to $k = 2$ relations in the body. We use a corpus of 1 million raw extractions, corresponding to 250,000 distinct facts. SHERLOCK found 5 million candidate rules that infer at least two of the observed facts. Unless otherwise noted, we use SHERLOCK's rule scoring function to evaluate the rules (Section 3.4).

The results represent a wide variety of domains, covering a total of 10,672 typed relations. We observe between a dozen and 2,375 distinct, ground facts for each relation. SHERLOCK learned a total of 31,000 inference rules.[2] Learning all rules, rule

[2] The learned rules are available at: http://www.cs.washington.edu/research/sherlock-hornclauses/

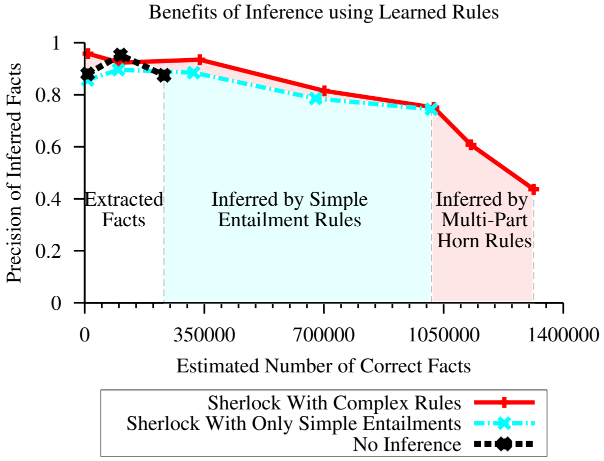


Figure 2: Inference discovers many facts which are not explicitly extracted, identifying 3x as many high quality facts (precision 0.8) and more than 5x as many facts overall. Horn-clauses with multiple relations in the body infer 30% more correct facts than are identified by simpler entailment rules, inferring many facts not present in the corpus in any form.

weights, and performing the inference took 50 minutes on a 72 core cluster. However, we note that for half of the relations SHERLOCK accepts no inference rules, and remind the reader that the performance on any particular relation may be substantially different, and depends on the facts observed in the corpus and on the rules learned.

4.2 Benefits of Inference

We first evaluate the utility of the learned Horn rules by contrasting the precision and number of correct and incorrect facts identified with and without inference over learned rules. We compare against two simpler variants of SHERLOCK. The first is a no-inference baseline that uses no rules, returning only facts that are explicitly extracted. The second baseline only accepts rules of length $k = 1$, allowing it to make simple entailments but not more complicated inferences using multiple facts.

Figure 2 compares the precision and estimated number of correct facts with and without inference. As is apparent, the learned inference rules substantially increase the number of known facts, quadrupling the number of correct facts beyond what are explicitly extracted.

The Horn rules having a body-length of two identify 30% more facts than the simpler length-one rules. Furthermore, we find the Horn rules yield slightly increased precision at comparable levels of recall, although the increase is not statistically significant. This behavior can be attributed to learning smaller weights for the length-two rules than the length-one rules, allowing the length-two rules provide a small amount of additional evidence as to which facts are true, but typically not enough to overcome the confidence of a more reliable length-one rule.

Analyzing the errors, we found that about one third of SHERLOCK's mistakes are due to metonymy and word sense ambiguity (*e.g.*, confusing Vancouver, British Columbia with Vancouver, Washington), one third are due to inferences based on incorrectly-extracted facts (*e.g.*, inferences based on the incorrect fact $\texttt{IsLocatedIn}(\texttt{New York}, \texttt{Suffolk County})$, which was extracted from sentences like 'Deer Park, New York is located in Suffolk County'), and the rest are due to unsound or incorrect inference rules (*e.g.*, $\texttt{BasedIn}(\texttt{Company}, \texttt{City})$:- $\texttt{BasedIn}(\texttt{Company}, \texttt{Country}) \wedge \texttt{CapitalOf}(\texttt{City}, \texttt{Country})$). Without negative examples it is difficult to distinguish correct rules from these unsound rules, since the unsound rules are correct more often than expected by chance.

Finally, we note that although simple, length-one rules capture many of the results, in some respects they are just rephrasing facts that are extracted in another form. However, the more complex, length-two rules synthesize facts extracted from multiple pages, and infer results that are not stated anywhere in the corpus.

4.3 Effect of Scoring Function

We now examine how SHERLOCK's rule scoring function affects its results, by comparing it with three rule scoring functions used in prior work:

LIME. The LIME ILP system (McCreath and Sharma, 1997) proposed a metric that generalized Muggleton's (1997) positive-only score function by modeling noise and limited sample sizes.

M-Estimate of rule precision. This is a common approach for handling noise in ILP (Dzeroski and Bratko, 1992). It requires negative examples, which we generated by randomly swapping arguments between positive examples.

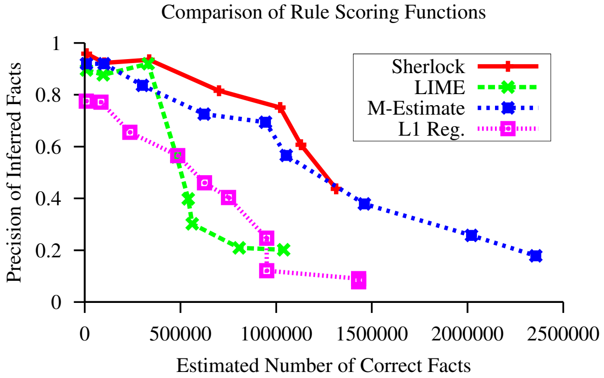


Figure 3: SHERLOCK identifies rules that lead to more accurate inferences over a large set of open-domain extracted facts, deducing 2x as many facts at precision 0.8.

L1 Regularization. As proposed in (Huynh and Mooney, 2008), this learns weights for all candidate rules using L_1-regularization (encouraging sparsity) instead of L_2-regularization, and retains only those with non-zero weight.

Figure 3 compares the precision and estimated number of correct facts inferred by the rules of each scoring function. SHERLOCK has consistently higher precision, and finds twice as many correct facts at precision 0.8.

M-Estimate accepted eight times as many rules as SHERLOCK, increasing the number of inferred facts at the cost of precision and longer inference times. Most of the errors in M-Estimate and L1 Regularization come from incorrect or unsound rules, whereas most of the errors for LIME stem from systematic extraction errors.

4.4 Scoring Function Design Decisions

SHERLOCK requires a rule to have statistical relevance and statistical significance. We perform an ablation study to understand how each of these contribute to SHERLOCK's results.

Figure 4 compares the precision and estimated number of correct facts obtained when requiring rules to be only statistically relevant, only statistically significant, or both. As is expected, there is a precision/recall tradeoff. SHERLOCK has higher precision, finding more than twice as many results at precision 0.8 and reducing the error by 39% at a recall of 1 million correct facts. Statistical significance finds twice as many correct facts as SHERLOCK, but the extra facts it discovers have precision < 0.4.

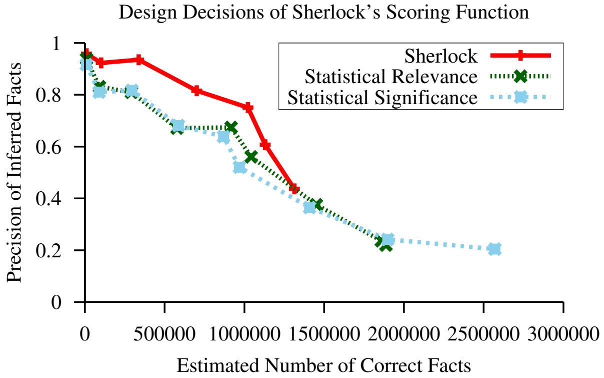


Figure 4: By requiring rules to have both statistical relevance and statistical significance, SHERLOCK rejects many error-prone rules that are accepted by the metrics individually. The better rule set yields more accurate inferences, but identifies fewer correct facts.

Comparing the rules accepted in each case, we found that statistical relevance and statistical significance each accepted about 180,000 rules, compared to about 31,000 for SHERLOCK. The smaller set of rules accepted by SHERLOCK not only leads to higher precision inferences, but also speeds up inference time by a factor of seven.

In a qualitative analysis, we found the statistical relevance metric overestimates probabilities for sparse rules, which leads to a number of very high scoring but meaningless rules. The statistical significance metric handles sparse rules better, but is still overconfident in the case of many unsound rules.

4.5 Analysis of Weight Learning

Finally, we empirically validate the modifications of the weight learning algorithm from Section 3.5.1.

The learned-rule weights only affect the probabilities of the inferred facts, not the inferred facts themselves, so to measure the influence of the weight learning algorithm we examine the recall at precision 0.8 and the area under the precision-recall curve (AuC). We build a test set by holding SHERLOCK's inference rules constant and randomly sampling 700 inferred facts. We test the effects of:

- Fixed *vs.* Variable Penalty - Do we use the same L_2 penalty on the weights for all rules or a stronger L_2 penalty for longer rules?
- Full *vs.* Weighted Grounding Counts - Do we count all unweighted rule groundings (as in (Huynh and Mooney, 2008)), or only the best weighted one (as in Equation 1)?

	Recall (p=0.8)	AuC
Variable Penalty, Weighted Counts (used by SHERLOCK)	**0.35**	**0.735**
Variable Penalty, Full Counts	0.28	0.726
Fixed Penalty, Weighted Counts	0.27	0.675
Fixed Penalty, Full Counts	0.17	0.488

Table 2: SHERLOCK's modified weight learning algorithm gives better probability estimates over noisy and incomplete Web extractions. Most of the gains come from penalizing longer rules more, but using weighted grounding counts further improves recall by 0.07, which corresponds to almost 100,000 additional facts at precision 0.8.

We vary each of these independently, and give the performance of all 4 combinations in Table 2.

The modifications from Section 3.5.1 improve both the AuC and the recall at precision 0.8. Most of the improvement is due to using stronger penalties on longer rules, but using the weighted counts in Equation 1 improves recall by a factor of 1.25 at precision 0.8. While this may not seem like much, the scale is such that it leads to almost 100,000 additional correct facts at precision 0.8.

5 Conclusion

This paper addressed the problem of learning first-order Horn clauses from the noisy and heterogeneous corpus of open-domain facts extracted from Web text. We showed that SHERLOCK is able to learn Horn clauses in a large-scale, domain-independent manner. Furthermore, the learned rules are valuable, because they infer a substantial number of facts which were not extracted from the corpus.

While SHERLOCK belongs to the broad category of ILP learners, it has a number of novel features that enable it to succeed in the challenging, open-domain context. First, SHERLOCK automatically identifies a set of high-quality extracted facts, using several simple but effective heuristics to defeat noise and ambiguity. Second, SHERLOCK is unsupervised and does not require negative examples; this enables it to scale to an unbounded number of relations. Third, it utilizes a novel rule-scoring function, which is tolerant of the noise, ambiguity, and missing data issues prevalent in facts extracted from Web text. The experiments in Figure 3 show that, for open-domain facts, SHERLOCK's method represents a substantial improvement over traditional ILP scoring functions.

Directions for future work include inducing longer inference rules, investigating better methods for combining the rules, allowing deeper inferences across multiple rules, evaluating our system on other corpora and devising better techniques for handling word sense ambiguity.

Acknowledgements

We thank Sonal Gupta and the anonymous reviewers for their helpful comments. This research was supported in part by NSF grant IIS-0803481, ONR grant N00014-08-1-0431, the WRF / TJ Cable Professorship and carried out at the University of Washington's Turing Center. The University of Washington gratefully acknowledges the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract nos. FA8750-09-C-0179 and FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

References

M. Banko, M. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction from the Web. In *Procs. of IJCAI*.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*.

M. Craven, D. DiPasquo, D. Freitag, A.K. McCallum, T. Mitchell, K. Nigam, and S. Slattery. 1998. Learning to Extract Symbolic Knowledge from the World Wide Web. In *Procs. of the 15th Conference of the American Association for Artificial Intelligence*, pages 509–516, Madison, US. AAAI Press, Menlo Park, US.

I. Dagan, O. Glickman, and B. Magnini. 2005. The PASCAL Recognising Textual Entailment Challenge. *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, pages 1–8.

S. Dzeroski and I. Bratko. 1992. Handling noise in inductive logic programming. In *Proceedings of the 2nd*

International Workshop on Inductive Logic Programming.

M. Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Procs. of the 14th International Conference on Computational Linguistics*, pages 539–545, Nantes, France.

T.N. Huynh and R.J. Mooney. 2008. Discriminative structure and parameter learning for Markov logic networks. In *Proceedings of the 25th international conference on Machine learning*, pages 416–423. ACM.

Stanley Kok and Pedro Domingos. 2005. Learning the structure of markov logic networks. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 441–448, New York, NY, USA. ACM.

N. Lavrac and S. Dzeroski, editors. 2001. *Relational Data Mining*. Springer-Verlag, Berlin, September.

D. Lin and P. Pantel. 2001. DIRT – Discovery of Inference Rules from Text. In *KDD*.

D. Lin and P. Pantel. 2002. Concept discovery from text. In *Proceedings of the 19th International Conference on Computational linguistics (COLING-02)*, pages 1–7.

E. McCreath and A. Sharma. 1997. ILP with noise and fixed example size: a Bayesian approach. In *Proceedings of the Fifteenth international joint conference on Artifical intelligence-Volume 2*, pages 1310–1315. Morgan Kaufmann Publishers Inc.

G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. 1990. Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–312.

S. Muggleton. 1995. Inverse entailment and Progol. *New Generation Computing*, 13:245–286.

S. Muggleton. 1997. Learning from positive data. *Lecture Notes in Computer Science*, 1314:358–376.

P. Pantel, R. Bhagat, B. Coppola, T. Chklovski, and E. Hovy. 2007. ISP: Learning inferential selectional preferences. In *Proceedings of NAACL HLT*, volume 7, pages 564–571.

M. Pennacchiotti and F.M. Zanzotto. 2007. Learning Shallow Semantic Rules for Textual Entailment. *Proceedings of RANLP 2007*.

J. R. Quinlan. 1990. Learning logical definitions from relations. *Machine Learning*, 5:239–2666.

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

M. Richardson and P. Domingos. 2006. Markov Logic Networks. *Machine Learning*, 62(1-2):107–136.

W.C. Salmon, R.C. Jeffrey, and J.G. Greeno. 1971. *Statistical explanation & statistical relevance*. Univ of Pittsburgh Pr.

S. Schoenmackers, O. Etzioni, and D. Weld. 2008. Scaling Textual Inference to the Web. In *Procs. of EMNLP*.

Y. Shinyama and S. Sekine. 2006. Preemptive information extraction using unrestricted relation discovery. In *Procs. of HLT/NAACL*.

R. Snow, D. Jurafsky, and A. Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *COLING/ACL 2006*.

M. Tatu and D. Moldovan. 2007. COGEX at RTE3. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 22–27.

M.P. Wellman, J.S. Breese, and R.P. Goldman. 1992. From knowledge bases to decision models. *The Knowledge Engineering Review*, 7(1):35–53.

A. Yates and O. Etzioni. 2007. Unsupervised resolution of objects and relations on the Web. In *Procs. of HLT*.

Constraints based Taxonomic Relation Classification

Quang Xuan Do **Dan Roth**
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
{quangdo2,danr}@illinois.edu

Abstract

Determining whether two terms in text have an ancestor relation (e.g. *Toyota* and *car*) or a sibling relation (e.g. *Toyota* and *Honda*) is an essential component of textual inference in NLP applications such as Question Answering, Summarization, and Recognizing Textual Entailment. Significant work has been done on developing stationary knowledge sources that could potentially support these tasks, but these resources often suffer from low coverage, noise, and are inflexible when needed to support terms that are not identical to those placed in them, making their use as general purpose background knowledge resources difficult. In this paper, rather than building a stationary hierarchical structure of terms and relations, we describe a system that, given two terms, determines the taxonomic relation between them using a machine learning-based approach that makes use of existing resources. Moreover, we develop a global constraint optimization inference process and use it to leverage an existing knowledge base also to enforce relational constraints among terms and thus improve the classifier predictions. Our experimental evaluation shows that our approach significantly outperforms other systems built upon existing well-known knowledge sources.

1 Introduction

Taxonomic relations that are read off of structured ontological knowledge bases have been shown to play important roles in many computational linguistics tasks, such as document clustering (Hotho et al., 2003), navigating text databases (Chakrabarti et al., 1997), Question Answering (QA) (Saxena et al., 2007) and summarization (Vikas et al., 2008). It is clear that the recognition of taxonomic relation between terms in sentences is essential to support textual inference tasks such as Recognizing Textual Entailment (RTE) (Dagan et al., 2006). For example, it may be important to know that a *blue Toyota* is neither a *red Toyota* nor a *blue Honda*, but that all are cars, and even Japanese cars. Work in Textual Entailment has argued quite convincingly (MacCartney and Manning, 2008; MacCartney and Manning, 2009) that many such textual inferences are largely compositional and depend on the ability to recognize some basic taxonomic relations such as the ancestor or sibling relations between terms. To date, these taxonomic relations can be read off manually generated ontologies such as Wordnet that explicitly represent these, and there has also been some work trying to extend the manually built resources using automatic acquisition methods resulting in structured knowledge bases such as the Extended WordNet (Snow et al., 2006) and the YAGO ontology (Suchanek et al., 2007).

However, identifying when these relations hold using fixed stationary hierarchical structures may be impaired by noise in the resource and by uncertainty in mapping targeted terms to concepts in the structures. In addition, for knowledge sources derived using bootstrapping algorithms and distributional semantic models such as (Pantel and Pennacchiotti, 2006; Kozareva et al., 2008; Baroni and Lenci, 2010), there is typically a trade-off between precision and recall, resulting either in a relatively accurate resource with low coverage or a noisy re-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1099–1109,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

source with broader coverage. In the current work, we take a different approach, identifying directly whether a pair of terms hold a taxonomic relation.

Fixed resources, as we observe, are inflexible when dealing with targeted terms not being covered. This often happens when targeted terms have the same meaning, but different surface forms, than the terms used in the resources (e.g. *Toyota Camry* and *Camry*). We argue that it is essential to have a classifier that, given two terms, can build a semantic representation of the terms and determines the taxonomic relations between them. This classifier will make use of existing knowledge bases in multiple ways, but will provide significantly larger coverage and more precise results. We make use of a dynamic resource such as Wikipedia to guarantee increased coverage without changing our model and also perform normalization-to-Wikipedia to find appropriate Wikipedia replacements for outside-Wikipedia terms. Moreover, stationary resources are usually brittle because of the way most of them are built: using local relational patterns (e.g. (Hearst, 1992; Snow et al., 2005)). Infrequent terms are less likely to be covered, and some relations may not be supported well by these methods because their corresponding terms rarely appear in close proximity (e.g., an Israeli tennis player *Dudi Sela* and *Roger Federrer*). Our approach uses search techniques to gather relevant Wikipedia pages of input terms and performs a learning-based classification w.r.t. to the features extracted from these pages as a way to get around this brittleness.

Motivated by the needs of NLP applications such as RTE, QA, Summarization, and the compositionality argument alluded to above, we focus on identifying two fundamental types of taxonomic relations - *ancestor* and *sibling*. An ancestor relation and its directionality can help us infer that a statement with respect to the child (e.g. *cannabis*) holds for an ancestor (e.g. *drugs*) as in the following example, taken from a textual entailment challenge dataset:

> **T**: Nigeria's NDLEA has seized 80 metric tonnes of *cannabis* in one of its largest ever hauls, officials say.
>
> **H**: Nigeria seizes 80 tonnes of *drugs*.

Similarly, it is important to know of a sibling relation to infer that a statement about *Taiwan* may (without additional information) contradict a similar statement with respect to *Japan* since these are different countries, as in the following:

> **T**: A strong earthquake struck off the southern tip of *Taiwan* at 12:26 UTC, triggering a warning from Japan's Meteorological Agency that a 3.3 foot tsunami could be heading towards Basco, in the Philippines.
>
> **H**: An earthquake strikes *Japan*.

Several recent TE studies (Abad et al., 2010; Sammons et al., 2010) suggest to isolate TE phenomena, such as recognizing taxonomic relations, and study them separately; they discuss some of characteristics of phenomena such as contradiction from a similar perspective to ours, but do not provide a solution.

In this paper, we present **TA**xonomic **RE**lation **C**lassifier (TAREC), a system that classifies taxonomic relations between a given pair of terms using a machine learning based classifier. An integral part of TAREC is also our inference model that makes use of relational constraints to enforce coherency among several related predictions. TAREC does not aim at building or extracting a hierarchical structure of concepts and relations, but rather to directly recognize taxonomic relations given a pair of terms. Target terms are represented using vector of features that are extracted from retrieved corresponding Wikipedia pages. In addition, we make use of existing stationary ontologies to find related terms to the target terms, and classify those too. This allows us to make use of a constraint-based inference model (following (Roth and Yih, 2004; Roth and Yih, 2007) that enforces coherency of decisions across related pairs (e.g., if x is-a y and y is-a z, it cannot be that x is a sibling of z).

In the rest of the paper, after discussing related work in Section 2, we present an overview of TAREC in Section 3. The learning component and the inference model of TAREC are described in Sections 4 and 5. We experimentally evaluate TAREC in Section 6 and conclude our paper in Section 7.

2 Related Work

There are several works that aim at building taxonomies and ontologies which organize concepts and their taxonomic relations into hierarchical structures. (Snow et al., 2005; Snow et al., 2006) constructed classifiers to identify hypernym relationship between terms from dependency trees of large corpora. Terms with recognized hypernym relation are extracted and incorporated into a man-made lexical database, WordNet (Fellbaum, 1998), resulting in the *extended WordNet*, which has been augmented with over $400,000$ synsets. (Ponzetto and Strube, 2007) and (Suchanek et al., 2007) both mined Wikipedia to construct hierarchical structures of concepts and relations. While the former exploited Wikipedia category system as a conceptual network and extracted a taxonomy consisting of subsumption relations, the latter presented the YAGO ontology, which was automatically constructed by mining and combining Wikipedia and WordNet. A natural way to use these hierarchical structures to support taxonomic relation classification is to map targeted terms onto the hierarchies and check if they subsume each other or share a common subsumer. However, this approach is limited because constructed hierarchies may suffer from noise and require exact mapping (Section 6). TAREC overcomes these limitations by searching and selecting the top relevant articles in Wikipedia for each input term; taxonomic relations are then recognized based on the features extracted from these articles.

On the other hand, information extraction bootstrapping algorithms, such as (Pantel and Pennacchiotti, 2006; Kozareva et al., 2008), automatically harvest related terms on large corpora by starting with a few seeds of pre-specified relations (e.g. *is-a*, *part-of*). Bootstrapping algorithms rely on some scoring function to assess the quality of terms and additional patterns extracted during bootstrapping iterations. Similarly, but with a different focus, Open IE, (Banko and Etzioni, 2008; Davidov and Rappoport, 2008), deals with a large number of relations which are not pre-specified. Either way, the output of these algorithms is usually limited to a small number of high-quality terms while sacrificing coverage (or vice versa). Moreover, an Open IE system cannot control the extracted relations and this is essential when identifying taxonomic relations. Recently, (Baroni and Lenci, 2010) described a general framework of distributional semantic models that extracts significant contexts of given terms from large corpora. Consequently, a term can be represented by a vector of contexts in which it frequently appears. Any vector space model could then use the terms' vectors to cluster terms into categories. Sibling terms (e.g. *Honda*, *Toyota*), therefore, have very high chance to be clustered together. Nevertheless, this approach cannot recognize ancestor relations. In this paper, we compare TAREC with this framework only on recognizing sibling vs. no relation, in a strict experimental setting which pre-specifies the categories to which the terms belong.

3 An Overview of the TAREC Algorithm

3.1 Preliminaries

In the TAREC algorithm, a *term* refers to any mention in text, such as *mountain*, *George W. Bush*, *battle of Normandy*. TAREC does not aim at extracting terms and building a stationary hierarchical structure of terms, but rather recognize the taxonomic relation between any two given terms. TAREC focuses on classifying two fundamental types of taxonomic relations: *ancestor* and *sibling*. Determining whether two terms hold a taxonomic relation depends on a pragmatic decision of how far one wants to climb up a taxonomy to find a common subsumer. For example, *George W. Bush* is a child of *Presidents of the United States* as well as *people*, even more, that term could also be considered as a child of *mammals* or *organisms* w.r.t. the Wikipedia category system; in that sense, *George W. Bush* may be considered as a sibling of *oak* because they have *organisms* as a least common subsumer. TAREC makes use of a hierarchical structure as background knowledge and considers two terms to hold a taxonomic relation only if the relation can be recognized from information acquired by climbing up at most K levels from the representation of the target terms in the structure. It is also possible that the sibling relation can be recognized by clustering terms together by using vector space models. If so, two terms are siblings if they belong to the same cluster.

To cast the problem of identifying taxonomic relations between two terms x and y in a machine learning perspective, we model it as a multi-class classification problem. Table 1 defines four relations with some examples in our experiment data sets.

This paper focuses on studying a fundamental problem of recognizing taxonomic relations (given well-segmented terms) and leaves the orthogonal is-

Relation	Meaning	Examples	
		Term x	Term y
$x \leftarrow y$	x is an ancestor of y	actor	Mel Gibson
		food	rice
$x \rightarrow y$	x is a child of y	Makalu	mountain
		Monopoly	game
$x \leftrightarrow y$	x and y are siblings	Paris	London
		copper	oxygen
$x \not\leftrightarrow y$	x and y have no relation	Roja	C++
		egg	Vega

Table 1: Taxonomic relations and some examples in our data sets.

```
TAxonomic RElation Classifier (TAREC)

INPUT:  A pair of terms $(x, y)$
        A learned local classifier $\mathcal{R}$ (Sec. 4)
        Wikipedia $\mathcal{W}$
OUTPUT: Taxonomic relation $r^*$ between $x$ and $y$
1.  $(x, y) \leftarrow$ NormalizeToWikipedia$(x, y, \mathcal{W})$
2.  $\mathcal{Z} \leftarrow$ GetAddionalTerms$(x, y)$ (Sec. 5.2)
3.  $r^* =$ ClassifyAndInference$(x, y, \mathcal{Z}, \mathcal{R}, \mathcal{W})$ (Sec. 5.1)
RETURN: $r^*$;
```

Figure 1: The TAREC algorithm.

sues of how to take contexts into account and how it should be used in applications to a future work.

3.2 The Overview of TAREC

Assume that we already have a learned local classifier that can classify taxonomic relations between any two terms. Given two terms, TAREC uses Wikipedia and the local classifier in an inference model to make a final prediction on the taxonomic relation between these two. To motivate the need for an inference model, beyond the local classifier itself, we observe that the presence of other terms in addition to the two input terms, can provide some natural constraints on the possible taxonomic relations and thus can be used to make the final prediction (which we also refer as global prediction) more coherent. In practice, we first train a local classifier (Section 4), then incorporate it into an inference model (Section 5) to classify taxonomic relations between terms. The TAREC algorithm consists of three steps and is summarized in Figure 1 and explained below.

1. Normalizing input terms to Wikipedia: Although most commonly used terms have corresponding Wikipedia articles, there are still a lot of terms with no corresponding Wikipedia articles. For a non-Wikipedia term, we make an attempt to find a replacement by using Web search. We wish to find a replacement such that the taxonomic relation is unchanged. For example, for input pair (*Lojze Kovačič*, *Rudi Šeligo*), there is no English Wikipedia page for *Lojze Kovačič*, but if we can find *Marjan Rožanc* and use it as a replacement of *Lojze Kovačič* (two terms are siblings and refer to two writers), we can continue classifying the taxonomic relation of the pair (*Marjan Rožanc*, *Rudi Šeligo*). This part of the algorithm was motivated by (Sarmento et al., 2007). We first make a query with the two input terms (e.g. "*Lojze Kovačič*" AND "*Rudi Šeligo*") to search for list-structure snippets in Web documents[1] such as "... $\langle delimiter \rangle$ c_a $\langle delimiter \rangle$ c_b $\langle delimiter \rangle$ c_c $\langle delimiter \rangle$ ..." (the two input terms should be among c_a, c_b, c_c, ...). The delimiter could be commas, periods, or asterisks[2]. For snippets that contain the patterns of interest, we extract c_a, c_b, c_c etc. as replacement candidates. To reduce noise, we empirically constrain the list to contain at least 4 terms that are no longer than 20 characters each. The candidates are ranked based on their occurrence frequency. The top candidate with Wikipedia pages is used as a replacement.

2. Getting additional terms (Section 5.2): TAREC leverage an existing knowledge base to extract additional terms related to the input terms, to be used in the inference model in step 3.

3. Making global prediction with relational constraints (Section 5.1): TAREC performs several local predictions using the local classifier $\mathcal{R}$ (Section 4) on the two input terms and these terms with the additional ones. The global prediction is then inferred by enforcing relational constraints among the terms' relations.

4 Learning Taxonomic Relations

The local classifier of TAREC is trained on the pairs of terms with correct taxonomic relation labels (some examples are showed in Table 1). The trained classifier when applied on a new input pair of terms will return a real valued number which can be interpreted as the probability of the predicted label. In this section, we describe the learning features used

[1] We use http://developer.yahoo.com/search/web/

[2] Periods and asterisks capture enumerations.

Title/Term	Text	Categories
President of the United States	*The President of the United States is the head of state and head of government of the United States and is the highest political official in the United States by influence and recognition. The President leads the executive branch of the federal government and is one of only two elected members of the executive branch...*	*Presidents of the United States, Presidency of the United States*
George W. Bush	*George Walker Bush; born July 6, 1946) served as the 43rd President of the United States from 2001 to 2009. He was the 46th Governor of Texas from 1995 to 2000 before being sworn in as President on January 20, 2001...*	*Children of Presidents of the United States, Governors of Texas, Presidents of the United States, Texas Republicans...*
Gerald Ford	*Gerald Rudolff Ford (born Leslie Lynch King, Jr.) (July 14, 1913 December 26, 2006) was the 38th President of the United States, serving from 1974 to 1977, and the 40th Vice President of the United States serving from 1973 to 1974.*	*Presidents of the United States, Vice Presidents of the United States, Republican Party (United States) presidential nominees...*

Table 2: Examples of texts and categories of Wikipedia articles.

by our local taxonomic relation classifier.

Given two input terms, we first build a semantic representation for each term by using a local search engine[3] to retrieve a list of top articles in Wikipedia that are relevant to the term. To do this, we use the following procedure: (1) Using both terms to make a query (e.g. *"George W. Bush"* AND *"Bill Clinton"*) to search in Wikipedia ; (2) Extracting important keywords in the titles and categories of the retrieved articles using TF-IDF (e.g. *president*, *politician*); (3) Combining each input term with the extracted keywords (e.g. *"George W. Bush"* AND *"president"* AND *"politician"*) to create a final query used to search for the term's relevant articles in Wikipedia. This is motivated by the assumption that the real world applications calling TAREC typically does so with two terms that are related in some sense, so our procedure is designed to exploit that. For example, it's more likely that term *Ford* in the pair (*George W. Bush*, *Ford*) refers to the former president of the United States, *Gerald Ford*, than the founder of Ford Motor Company, *Henry Ford*.

Once we have a semantic representation of each term, in the form of the extracted articles, we extract from it features that we use as the representation of the two input terms in our learning algorithm. It is worth noting that a Wikipedia page usually consists of a title (i.e. the term), a body text, and a list of categories to which the page belongs. Table 2 shows some Wikipedia articles. From now on, we use *the titles of* x, *the texts of* x, and *the categories of* x to refer to the titles, texts, and categories of the associated articles in the representation of x. Below are the learning features extracted for input pair (x,y).

Bags-of-words Similarity: We use cosine similarity metric to measure the degree of similarity between bags of words. We define four bags-of-words features as the degree of similarity between the texts and categories associated with two input terms x and y in Table 3. To collect categories of a term, we take the categories of its associated articles and go up K levels in the Wikipedia category system. In our experiments, we use abstracts of Wikipedia articles instead of whole texts.

Degree of similarity
texts(x) vs. categories(y)
categories(x) vs. texts(y)
texts(x) vs. texts(y)
categories(x) vs. categories(y)

Table 3: Bag-of-word features of the pair of terms (x,y); texts(.) and categories(.) are two functions that extract associated texts and categories from the semantic representation of x and y.

Association Information: This features represents a measure of association between the terms by considering their information overlap. We capture this feature by the pointwise mutual information (pmi) which quantifies the discrepancy between the probability of two terms appearing together versus the probability of each term appearing independently[4]. The pmi of two terms x and y is estimated as follows:

$$pmi(x, y) = log\frac{p(x, y)}{p(x)p(y)} = log\frac{Nf(x, y)}{f(x)f(y)},$$

where N is the total number of Wikipedia articles, and $f(.)$ is the function which counts the number of appearances of its argument.

Overlap Ratios: The overlap ratio features capture the fact that the titles of a term usually overlap with the categories of its descendants. We measure this overlap as the ratio of the number of common phrases used in the titles of one term and the categories of the other term. In our context, a phrase is

[3] E.g. http://lucene.apache.org/

[4] pmi is different than mutual information. The former applies to specific outcomes, while the latter is to measure the mutual dependence of two random variables.

considered to be a common phrase if it appears in the titles of one term and the categories of the other term and it is also of the following types: (1) the whole string of a category, or (2) the head in the root form of a category, or (3) the post-modifier of a category. We use the Noun Group Parser from (Suchanek et al., 2007) to extract the head and post-modifier from a category. For example, one of the categories of an article about *Chicago* is *Cities in Illinois*. This category can be parsed into a head in its root form *City*, and a post-modifier *Illinois*. Given term pair *(City, Chicago)*, we observe that *City* matches the head of the category *Cities in Illinois* of term *Chicago*. This is a strong indication that *Chicago* is a child of *City*.

We also use a feature that captures the overlap ratio of *common phrases* between the categories of two input terms. For this feature, we do not use the *post-modifier* of the categories. We use Jaccard similarity coefficient to measure these overlaps ratios.

5 Inference with Relational Constraints

Once we have a local multi-class classifier that maps a given pair of terms to one of the four possible relations, we use a constraint-based optimization algorithm to improve this prediction. The key insight behind the way we model the inference model is that if we consider more than two terms, there are logical constraints that restrict the possible relations among them. For instance, *George W. Bush* cannot be an ancestor or sibling of *president* if we are confident that *president* is an ancestor of *Bill Clinton*, and *Bill Clinton* is a sibling of *George W. Bush*. We call the combination of terms and their relations a *term network*. Figure 2 shows some n-term networks consisting of two input terms (x, y), and additional terms z, w, v.

The aforementioned observations show that if we can obtain additional terms that are related to the two target terms, we can enforce such coherency relational constraints and make a global prediction that would improve the prediction of the taxonomic relation between the two given terms. Our inference model follows constraint-based formulations that were introduced in the NLP community and were shown to be very effective in exploiting declarative background knowledge (Roth and Yih, 2004; Denis and Baldridge, 2007; Punyakanok et al., 2008; Chang et al., 2008).

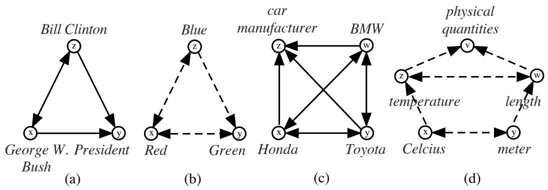


Figure 2: Examples of n-term networks with two input term x and y. (a) and (c) show valid combinations of edges, whereas (b) and (d) are two relational constraints. For simplicity, we do not draw *no relation* edges in (d).

5.1 Enforcing Coherency through Inference

Let x, y be two input terms, and $\mathcal{Z} = \{z_1, z_2, ..., z_m\}$ be a set of additional terms. For a subset $Z \in \mathcal{Z}$, we construct a set of term networks whose nodes are x, y and all elements in Z, and the edge, e, between every two nodes is one of four taxonomic relations whose weight, $w(e)$, is given by a local classifier (Section 4). If $l = |Z|$, there are $n = 2 + l$ nodes in each network, and $4^{\left[\frac{1}{2}n(n-1)\right]}$ term networks can be constructed. In our experiments we only use 3-term networks (i.e. $l = 1$). For example, for the input pair (*red*, *green*) and $\mathcal{Z} = \{blue, yellow\}$, we can construct 64 networks for the triple $\langle red, green, Z = \{blue\}\rangle$ and 64 networks for $\langle red, green, Z = \{yellow\}\rangle$ by trying all possible relations between the terms.

A *relational constraint* is defined as a term network consisting of only its "illegitimate" edge settings, those that belongs to a pre-defined list of invalid edge combinations. For example, Figure 2b shows an invalid network where *red* is a sibling of both *green* and *blue*, and *green* is an ancestor of *blue*. In Figure 2d, *Celcius* and *meter* cannot be siblings because they are children of two sibling terms *temperature* and *length*. The relational constraints used in our experiments are manually constructed.

Let $\mathcal{C}$ be a list of relational constraints. Equation (1) defines the network scoring function, which is a linear combination of the edge weights, $w(e)$, and the penalties, ρ_k, of term networks matching constraint $C_k \in \mathcal{C}$.

$$score(t) = \sum_{e \in t} w(e) - \sum_{k=1}^{|\mathcal{C}|} \rho_k d_{C_k}(t) \qquad (1)$$

function $d_{C_k}(t)$ indicates if t matches C_k. In our work, we use relational constraints as hard con-

YAGO Query Patterns

INPUT: term "x"

OUTPUT: lists of ancestors, siblings, and children of "x"

Pattern 1	Pattern 2	Pattern 3
"x" MEANS ?A	"x" MEANS ?A	"x" MEANS ?D
?A SUBCLASSOF ?B	?A TYPE ?B	?E TYPE ?D
?C SUBCLASSOF ?B	?C TYPE ?B	

RETURN: ?B, ?C, ?E as
lists of ancestors, siblings, and children, respectively.

Figure 3: Our YAGO query patterns used to obtain related terms for "x".

straints and set their penalty ρ_k to ∞. For a set of term networks formed by $\langle x, y, Z\rangle$ and all possible relations between the terms, we select the best network, $t^* = \text{argmax}_t score(t)$.

After picking the best term network t^* for every $Z \in \mathcal{Z}$, we make the final decision on the taxonomic relation between x and y. Let r denote the relation between x and y in a particular t^* (e.g. $r = x \leftrightarrow y$.) The set of all t^* is divided into 4 groups with respect to r (e.g. a group of all t^* having $r = x \leftrightarrow y$, a group of all t^* having $r = x \leftarrow y$.) We denote a group with term networks holding r as the relation between x and y by $\mathcal{T}_r$. To choose the best taxonomic relation, r^*, of x and y, we solve the objective function defined in Equation 2.

$$r^* = \text{argmax}_r \frac{1}{|\mathcal{T}_r|} \sum_{t^* \in \mathcal{T}_r} \lambda_{t^*} score(t^*) \qquad (2)$$

where λ_t is the weight of term network t, defined as the occurrence probability of t (regarding only its edges' setting) in the training data, which is augmented with additional terms. Equation (2) finds the best taxonomic relation of two input terms by computing the average score of every group of the best term networks representing a particular relation of two input terms.

5.2 Extracting Related Terms

In the inference model, we need to obtain other terms that are related to the two input terms. Hereafter, we refer to additional terms as *related terms*. The related term space is a space of direct ancestors, siblings and children in a particular resource.

We propose an approach that uses the YAGO ontology (Suchanek et al., 2007) to provide related terms. It is worth noting that YAGO is chosen over the Wikipedia category system used in our work because YAGO is a clean ontology built by carefully combining Wikipedia and WordNet.[5]

In YAGO model, all objects (e.g. *cities*, *people*, etc.) are represented as *entities*. To map our input terms to entities in YAGO, we use the MEANS relation defined in the YAGO ontology. Furthermore, similar entities are grouped into *classes*. This allows us to obtain direct ancestors of an entity by using the TYPE relation which gives the entity's classes. Furthermore, we can get ancestors of a class with the SUBCLASSOF relation[6]. By using three relations MEANS, TYPE and SUBCLASSOF in YAGO model, we can obtain Proposals for direct ancestors, siblings, and children, if any, for any input term. We then evaluate our classifier on all pairs, and run the inference to improve the prediction using the coherency constraints. Figure 3 presents three patterns that we used to query related terms from YAGO.

6 Experimental Study

In this section, we evaluate TAREC against several systems built upon existing well-known knowledge sources. The resources are either hierarchical structures or extracted by using distributional semantic models. We also perform several experimental analyses to understand TAREC's behavior in details.

6.1 Comparison to Hierarchical Structures

We create and use two main data sets in our experiments. **Dataset-I** is generated from 40 semantic classes of about 11,000 instances. The original semantic classes and instances were manually constructed with a limited amount of manual post-filtering and were used to evaluate information extraction tasks in (Paşca, 2007; Paşca and Van Durme, 2008) (we refer to this original data as **OrgData-I**). This dataset contains both terms with Wikipedia pages (e.g. *George W. Bush*) and non-Wikipedia terms (e.g. *hindu mysticism*). Pairs of terms are generated by randomly pairing semantic class names and instances. We generate disjoint training and test sets of 8,000 and 12,000 pairs of terms, respectively. We call the test set of this

[5] However, YAGO by itself is weaker than our approach in identifying taxonomic relations (see Section 6.)

[6] These relations are defined in the YAGO ontology.

dataset **Test-I**. **Dataset-II** is generated from 44 semantic classes of more than 10,000 instances used in (Vyas and Pantel, 2009)[7]. The original semantic classes and instances were extracted from Wikipedia lists. This data, therefore, only contains terms with corresponding Wikipedia pages. We also generate disjoint training and test sets of 8,000 and 12,000 pairs of terms, respectively, and call the test set of this dataset **Test-II**.[8]

Several semantic class names in the original data are written in short forms (e.g. *chemicalelem*, *proglanguage*). We expand these names to some meaningful names which are used by all systems in our experiments. For example, *terroristgroup* is expanded to *terrorist group*, *terrorism*. Table 1 shows some pairs of terms which are generated. Four types of taxonomic relations are covered with balanced numbers of examples in all data sets. To evaluate our systems, we use a snapshot of Wikipedia from July, 2008. After cleaning and removing articles without categories (except redirect pages), 5,503,763 articles remain. We index these articles using Lucene[9]. As a learning algorithm, we use a regularized averaged Perceptron (Freund and Schapire, 1999).

We compare TAREC with three systems that we built using recently developed large-scale hierarchical structures. **Strube07** is built on the latest version of a taxonomy, T_{Strube}, which was derived from Wikipedia (Ponzetto and Strube, 2007). It is worth noting that the structure of T_{Strube} is similar to the page structure of Wikipedia. For a fair comparison, we first generate a semantic representation for each input term by following the same procedure used in TAREC described in Section 4. The titles and categories of the articles in the representation of each input term are then extracted. Only titles and their corresponding categories that are in T_{Strube} are considered. A term is an ancestor of the other if at least one of its titles is in the categories of the other term. If two terms share a common category, they are considered siblings; and no relation, otherwise. The ancestor relation is checked first, then sibling, and finally no relation. **Snow06** uses the *extended WordNet* (Snow et al., 2006). Words in the extended WordNet can be common nouns or proper nouns. Given two input terms, we first map them onto the hierarchical structure of the extended WordNet by exact string matching. A term is an ancestor of the other if it can be found as an hypernym after going up K levels in the hierarchy from the other term. If two terms share a common subsumer within some levels, then they are considered as siblings. Otherwise, there is no relation between the two input terms. Similar to Strube07, we first check ancestor, then sibling, and finally no relation. **Yago07** uses the YAGO ontology (Suchanek et al., 2007) as its main source of background knowledge. Because the YAGO ontology is a combination of Wikipedia and WordNet, this system is expected to perform well at recognizing taxonomic relations. To access a term's ancestors and siblings, we use patterns 1 and 2 in Figure 3 to map a term to the ontology and move up on the ontology. The relation identification process is then similar to those of Snow06 and Strube07. If an input term is not recognized by these systems, they return *no relation*.

	Test-I	**Test-II**
Strube07	24.32	25.63
Snow06	41.97	36.26
Yago07	65.93	70.63
TAREC (local)	81.89	84.7
TAREC	**85.34**	**86.98**

Table 4: Evaluating and comparing performances, in accuracy, of the systems on **Test-I** and **Test-II**. TAREC (local) uses only our local classifier to identify taxonomic relations by choosing the relation with highest confidence.

Our overall algorithm, **TAREC**, is described in Figure 1. We manually construct a pre-defined list of 35 relational constraints to use in the inference model. We also evaluate our local classifier (Section 4), which is referred as **TAREC (local)**. To make classification decision with TAREC (local), for a pair of terms, we choose the predicted relation with highest confidence returned by the classifier.

In all systems compared, we vary the value of K[10] from 1 to 4. The best result of each system is reported. Table 4 shows the comparison of all systems evaluated on both Test-I and Test-II. Our systems, as shown, significantly outperform the other

[7] There were 50 semantic classes in the original dataset. We grouped some semantically similar classes for the purpose of classifying taxonomic relations.

[8] Published at http://cogcomp.cs.illinois.edu/page/software

[9] http://lucene.apache.org, version 2.3.2

[10] See Section 3.1 for the meaning of K.

systems. In Table 4, the improvement of TAREC over TAREC (local) on Test-I shows the contribution of both the normalization procedure (that is, going outside Wikipedia terms) and the global inference model to the classification decisions, whereas the improvement on Test-II shows only the contribution of the inference model, because Test-II contains only terms with corresponding Wikipedia articles.

Observing the results we see that our algorithms is doing significantly better that fixed taxonomies based algorithms. This is true both for TAREC (local) and for TAREC. We believe that our machine learning based classifier is very flexible in extracting features of the two input terms and thus in predicting their taxonomic Relation. On the other hand, other system rely heavily on string matching techniques to map input terms to their respective ontologies, and these are very inflexible and brittle. This clearly shows one limitation of using existing structured resources to classify taxonomic relations.

We do not use special tactics to handle polysemous terms. However, our procedure of building semantic representations for input terms described in Section 4 ties the senses of the two input terms and thus, implicitly, may get some sense information. We do not use this procedure in Snow06 because WordNet and Wikipedia are two different knowledge bases. We also do not use this procedure in Yago07 because in YAGO, a term is mapped onto the ontology by using the MEANS operator (in Pattern 1, Figure 3). This cannot follow our procedure.

6.2 Comparison to Harvested Knowledge

As we discussed in Section 2, the output of bootstrapping-based algorithms is usually limited to a small number of high-quality terms while sacrificing coverage (or vice versa). For example, the full Espresso algorithm in (Pantel and Pennacchiotti, 2006) extracted 69,156 instances of *is-a* relation with 36.2% precision. Similarly, (Kozareva et al., 2008) evaluated only a small number (a few hundreds) of harvested instances. Recently, (Baroni and Lenci, 2010) proposed a general framework to extract properties of input terms. Their **TypeDM** model harvested 5,000 significant properties for each term out of 20,410 noun terms. For example, the properties of *marine* include $\langle own, bomb\rangle$, $\langle use, gun\rangle$. Using vector space models we could measure the similarity between terms using their property vectors. However, since the information available in TypeDM does not support predicting the ancestor relation between terms, we only evaluate TypeDM in classifying sibling vs. no relation. We do this by giving a list of semantic classes using the following procedure: (1) For each semantic class, use some seeds to compute a centroid vector from the seeds' vectors in TypeDM, (2) each term in an input pair is classified into its best semantic class based on the cosine similarity between its vector and the centroid vector of the category, (3) two terms are siblings if they are classified into the same category; and have no relation, otherwise. Out of 20,410 noun terms in TypeDM, there are only 345 terms overlapping with the instances in OrgData-I and belonging to 10 significant semantic classes. For each semantic class, we randomly pick 5 instances as its seeds to make a centroid vector. The rest of the overlapping instances are randomly paired to make a dataset of 4,000 pairs of terms balanced in the number of sibling and no relation pairs. On this dataset, TypeDM achieves the accuracy of 79.75%. TAREC (local), with the local classifier trained on the training set (with 4 relation classes) of Dataset-I, gives 78.35% of accuracy. The full TAREC system with relational constraints achieves 82.65%. We also re-train and evaluate the local classifier of TAREC on the same training set but without ancestor relation pairs. This local classifier has an accuracy of 81.08%.

These results show that although the full TAREC system gives better performance, TypeDM is very competitive in recognizing sibling vs. no relation. However, TypeDM can only work in a limited setting where semantic classes are given in advance, which is not practical in real-world applications; and of course, TypeDM does not help to recognize ancestor relations between two terms.

6.3 Experimental Analysis

In this section, we discuss some experimental analyses to better understand our systems.

Precision and Recall: We want to study TAREC on individual taxonomic relations using Precision and Recall. Table 5 shows that TAREC performs very well on ancestor relation. Sibling and no relation are the most difficult relations to classify. In the same experimental setting on Test-I, Yago07

TAREC	Test-I		Test-II	
	Prec	Rec	Prec	Rec
$x \leftarrow y$	95.82	88.01	96.46	88.48
$x \rightarrow y$	94.61	89.29	96.15	88.86
$x \leftrightarrow y$	79.23	84.01	83.15	81.87
$x \not\leftrightarrow y$	73.94	79.9	75.54	88.27
Average	85.9	85.3	87.83	86.87

Table 5: Performance of TAREC on individual taxonomic relation.

	Wiki	**WordNet**	**non-Wiki**
Strube07	24.59	24.13	21.18
Snow06	41.23	46.91	34.46
Yago07	69.95	70.42	34.26
TAREC (local)	89.37	89.72	31.22
TAREC	**91.03**	**91.2**	**45.21**

Table 6: Performance of the systems on special data sets, in accuracy. On the non-Wikipedia test set, TAREC (local) simply returns sibling relation.

achieves 79.34% and 66.03% of average Precision and Recall, respectively. These numbers on Test-II are 81.33% and 70.44%.

Special Data Sets: We evaluate all systems that use hierarchical structures as background knowledge on three special data sets derived from Test-I. From 12,000 pairs in Test-I, we created a test set, **Wiki**, consisting of $10,456$ pairs with all terms in Wikipedia. We use the rest of $1,544$ pairs with at least one non-Wikipedia term to build a **non-Wiki** test set. The third dataset, **WordNet**, contains $8,625$ pairs with all terms in WordNet and Wikipedia. Table 6 shows the performance of the systems on these data sets. Unsurprisingly, Yago07 gets better results on Wiki than on Test-I. Snow06, as expected, gives better performance on the WordNet test set. TAREC still significantly outperforms these systems. The improvement of TAREC over TAREC (local) on the Wiki and WordNet test sets shows the contribution of the inference model, whereas the improvement on the non-Wikipedia test set shows the contribution of normalizing input terms to Wikipedia.

Contribution of Related Terms in Inference: We evaluate TAREC when the inference procedure is fed by related terms that are generated using a "gold standard" source instead of YAGO. To do this, we use the original data which was used to generate Test-I. For each term in the examples of Test-I, we get its ancestors, siblings, and children, if any, from the original data and use them as related terms in the inference model. This system is referred as **TAREC (Gold Infer.)**. Table 7 shows the results of the two systems on different K as the number of levels to go up on the Wikipedia category system. We see that TAREC gets better results when doing inference with better related terms. In this experiment, the two systems use the same number of related terms.

	K=1	K=2	K=3	K=4
TAREC	82.93	85.34	85.23	83.95
TAREC (Gold Infer.)	83.46	86.18	85.9	84.93

Table 7: Evaluating TAREC with different sources providing related terms to do inference.

7 Conclusions

We studied an important component of many computational linguistics tasks: given two target terms, determine that taxonomic relation between them. We have argued that static structured knowledge bases cannot support this task well enough, and provided empirical support for this claim. We have developed TAREC, a novel algorithm that leverages information from existing knowledge sources and uses machine learning and a constraint-based inference model to mitigate the noise and the level of uncertainty inherent in these resources. Our evaluations show that TAREC significantly outperforms other systems built upon existing well-known knowledge sources. Our approach generalizes and handles non-Wikipedia term well across semantic classes. Our future work will include an evaluation of TAREC in the context of textual inference applications.

Acknowledgments

The authors thank Mark Sammons, Vivek Srikumar, James Clarke and the anonymous reviewers for their insightful comments and suggestions. University of Illinois at Urbana-Champaign gratefully acknowledges the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract No. FA8750-09-C-0181. The first author also thanks the Vietnam Education Foundation (VEF) for its sponsorship. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the VEF, DARPA, AFRL, or the US government.

References

A. Abad, L. Bentivogli, I. Dagan, D. Giampiccolo, S. Mirkin, E. Pianta, and A. Stern. 2010. A resource for investigating the impact of anaphora and coreference on inference. In *LREC*.

M. Banko and O. Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *ACL-HLT*.

M. Baroni and A. Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36.

S. Chakrabarti, B. Dom, R. Agrawal, and P. Raghavan. 1997. Using taxonomy, discriminants, and signatures for navigating in text databases. In *VLDB*.

M. Chang, L. Ratinov, and D. Roth. 2008. Constraints as prior knowledge. In *ICML Workshop on Prior Knowledge for Text and Language Processing*.

D. Davidov and A. Rappoport. 2008. Unsupervised discovery of generic relationships using pattern clusters and its evaluation by automatically generated sat analogy questions. In *ACL*.

P. Denis and J. Baldridge. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. In *NAACL*.

C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Y. Freund and R. E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*.

M. A. Hearst. 1992. Acquisition of hyponyms from large text corpora. In *COLING*.

A. Hotho, S. Staab, and G. Stumme. 2003. Ontologies improve text document clustering. In *ICDM*.

Z. Kozareva, E. Riloff, and E. Hovy. 2008. Semantic class learning from the web with hyponym pattern linkage graphs. In *ACL-HLT*.

B. MacCartney and C. D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *COLING*.

B. MacCartney and C. D. Manning. 2009. An extended model of natural logic. In *IWCS-8*.

M. Paşca and B. Van Durme. 2008. Weakly-supervised acquisition of open-domain classes and class attributes from web documents and query logs. In *ACL-HLT*.

M. Paşca. 2007. Organizing and searching the world wide web of facts step two: Harnessing the wisdom of the crowds. In *WWW*.

P. Pantel and M. Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *ACL*, pages 113–120.

S. P. Ponzetto and M. Strube. 2007. Deriving a large scale taxonomy from wikipedia. *AAAI*.

V. Punyakanok, D. Roth, and W. Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2).

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *CoNLL*.

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.

M. Sammons, V.G. Vydiswaran, and D. Roth. 2010. Ask not what textual entailment can do for you... In *ACL*.

L. Sarmento, V. Jijkuon, M. de Rijke, and E. Oliveira. 2007. "more like these": growing entity classes from seeds. In *CIKM*.

A. K. Saxena, G. V. Sambhu, S. Kaushik, and L. V. Subramaniam. 2007. Iitd-ibmirl system for question answering using pattern matching, semantic type and semantic category recognition. In *TREC*.

R. Snow, D. Jurafsky, and A. Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *NIPS*.

R. Snow, D. Jurafsky, and A. Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *ACL*.

F. M. Suchanek, G. Kasneci, and G. Weikum. 2007. Yago: A Core of Semantic Knowledge. In *WWW*.

O. Vikas, A. K. Meshram, G. Meena, and A. Gupta. 2008. Multiple document summarization using principal component analysis incorporating semantic vector space model. In *Computational Linguistics and Chinese Language Processing*.

V. Vyas and P. Pantel. 2009. Semi-automatic entity set refinement. In *NAACL-HLT*.

D. Yarowsky. 1995. Unsupervised woed sense disambiguation rivaling supervied methods. In *Proceedings of ACL-95*.

A Semi-Supervised Method to Learn and Construct Taxonomies using the Web

Zornitsa Kozareva and Eduard Hovy
USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695
{kozareva,hovy}@isi.edu

Abstract

Although many algorithms have been developed to harvest lexical resources, few organize the mined terms into taxonomies. We propose (1) a semi-supervised algorithm that uses a root concept, a basic level concept, and recursive surface patterns to learn automatically from the Web hyponym-hypernym pairs subordinated to the root; (2) a Web based concept positioning procedure to validate the learned pairs' is-a relations; and (3) a graph algorithm that derives from scratch the integrated taxonomy structure of all the terms. Comparing results with WordNet, we find that the algorithm misses some concepts and links, but also that it discovers many additional ones lacking in WordNet. We evaluate the taxonomization power of our method on reconstructing parts of the WordNet taxonomy. Experiments show that starting from scratch, the algorithm can reconstruct 62% of the WordNet taxonomy for the regions tested.

1 Introduction

A variety of NLP tasks, including inference, textual entailment (Glickman et al., 2005; Szpektor et al., 2008), and question answering (Moldovan et al., 1999), rely on semantic knowledge derived from term taxonomies and thesauri such as WordNet. However, the coverage of WordNet is still limited in many regions (even well-studied ones such as the concepts and instances below Animals and People), as noted by researchers such as (Pennacchiotti and Pantel, 2006) and (Hovy et al., 2009) who perform automated semantic class learning. This happens because WordNet and most other existing taxonomies are manually created, which makes them difficult to maintain in rapidly changing domains, and (in the face of taxonomic complexity) makes them hard to build with consistency. To surmount these problems, it would be advantageous to have an automatic procedure that can not only augment existing resources but can also produce taxonomies for existing and new domains and tasks starting from scratch.

The main stages of automatic taxonomy induction are term extraction and term organization. In recent years there has been a substantial amount of work on term extraction, including semantic class learning (Hearst, 1992; Riloff and Shepherd, 1997; Etzioni et al., 2005; Pasca, 2004; Kozareva et al., 2008), relation acquisition between entities (Girju et al., 2003; Pantel and Pennacchiotti, 2006; Davidov et al., 2007), and creation of concept lists (Katz and Lin, 2003). Various attempts have been made to learn the taxonomic organization of concepts (Widdows, 2003; Snow et al., 2006; Yang and Callan, 2009). Among the most common is to start with a good ontology and then to try to position the missing concepts into it. (Snow et al., 2006) maximize the conditional probability of hyponym-hypernym relations given certain evidence, while (Yang and Callan, 2009) combines heterogenous features like context, co-occurrence, and surface patterns to produce a more-inclusive inclusion ranking formula. The obtained results are promising, but the problem of how to organize the gathered knowledge when there is no initial taxonomy, or when the initial taxonomy is grossly impoverished, still remains.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1110–1118,
MIT, Massachusetts, USA, 9-11 October 2010.

The major problem in performing taxonomy construction from scratch is that overall concept positioning is not trivial. It is difficult to discover whether concepts are unrelated, subordinated, or parallel to each other. In this paper, we address the following question: *How can one induce the taxonomic organization of concepts in a given domain starting from scratch?*

The contributions of this paper are as follows:

- An automatic procedure for harvesting hyponym-hypernym pairs given a domain of interest.
- A ranking mechanism for validating the learned is-a relations between the pairs.
- A graph-based approach for inducing the taxonomic organization of the harvested terms starting from scratch.
- An experiment on reconstructing WordNet's taxonomy for given domains.

Before focusing on the harvesting and taxonomy induction algorithms, we are going to describe some basic terminology following (Hovy et al., 2009). A **term** is an English word (for our current purposes, a noun or a proper name). A **concept** is an item in the classification taxonomy we are building. A **root concept** is a fairly general concept which is located on the high level of the taxonomy. A **basic-level concept** corresponds to the Basic Level categories defined in Prototype Theory in Psychology (Rosch, 1978). For example, a *dog*, not a mammal or a collie. An **instance** is an item in the classification taxonomy that is more specific than a concept. For example, *Lassie*, not a dog or collie .

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 describes the taxonomization framework. Section 4 discusses the experiments. We conclude in Section 5.

2 Related Work

The first stage of automatic taxonomy induction, term and relation extraction, is relatively well-understood. Methods have matured to the point of achieving high accuracy (Girju et al., 2003; Pantel and Pennacchiotti, 2006; Kozareva et al., 2008). The produced output typically contains flat lists of terms and/or ground instance facts ($lion$ is-a $mammal$) and general relation types ($mammal$ is-a $animal$).

Most approaches use either clustering or patterns to mine knowledge from structured and unstructured text. Clustering approaches (Lin, 1998; Lin and Pantel, 2002; Davidov and Rappoport, 2006) are fully unsupervised and discover relations that are not directly expressed in text. Their main drawback is that they may or may not produce the term types and granularities useful to the user. In contrast, pattern-based approaches harvest information with high accuracy, but they require a set of seeds and surface patterns to initiate the learning process. These methods are successfully used to collect semantic lexicons (Riloff and Shepherd, 1997; Etzioni et al., 2005; Pasca, 2004; Kozareva et al., 2008), encyclopedic knowledge (Suchanek et al., 2007), concept lists (Katz and Lin, 2003), and relations between terms, such as hypernyms (Ritter et al., 2009; Hovy et al., 2009) and part-of (Girju et al., 2003; Pantel and Pennacchiotti, 2006).

However, simple term lists are not enough to solve many problems involving natural language. Terms may be augmented with information that is required for knowledge-intensive tasks such as textual entailment (Glickman et al., 2005; Szpektor et al., 2008) and question answering (Moldovan et al., 1999). To support inference, (Ritter et al., 2010) learn the selectional restrictions of semantic relations, and (Pennacchiotti and Pantel, 2006) ontologize the learned arguments using WordNet.

Taxonomizing the terms is a very powerful method to leverage added information. Subordinated terms (hyponyms) inherit information from their superordinates (hypernyms), making it unnecessary to learn all relevant information over and over for every term in the language. But despite many attempts, no 'correct' taxonomization has ever been constructed for the terms of, say, English. Typically, people build term taxonomies (and/or richer structures like ontologies) for particular purposes, using specific taxonomization criteria. Different tasks and criteria produce different taxonomies, even when using the same basic level concepts. This is because most basic level concepts admit to multiple perspectives, while each task focuses on one, or at most two, perspectives at a time. For example, a dolphin is a Mammal (and not a Fish) to a biologist, but is a Fish (and hence not a Mammal) to a fisherman or anyone building or visiting an aquarium. More confusingly, a tiger and a puppy are both Mammals and hence belong close together in a typical taxonomy, but a tiger is a WildAnimal (in the perspective of Animal-Function) and a JungleDweller (in the perspective of Habitat), while a puppy is a Pet (as function) and a HouseAnimal (as habitat), which would place them relatively far from one another. Attempts at producing a single multi-perspective taxonomy fail due to the complexity of interaction among perspectives, and people are notoriously bad at constructing taxonomies adherent to a single perspective when given terms from multiple perspectives. This issue and the major alternative principles for taxonomization are discussed in (Hovy, 2002).

It is therefore not surprising that the second stage of automated taxonomy induction is harder to achieve. As mentioned, most attempts to learn taxonomy structures start with a reasonably complete taxonomy and then insert the newly learned terms into it, one term at a time (Widdows, 2003; Pasca, 2004; Snow et al., 2006; Yang and Callan, 2009). (Snow et al., 2006) guide the incremental approach by maximizing the conditional probability over a set of relations. (Yang and Callan, 2009) introduce a taxonomy induction framework which combines the power of surface patterns and clustering through combining numerous heterogeneous features.

Still, one would like a procedure to organize the harvested terms into a taxonomic structure starting fresh (i.e., without using an initial taxonomic structure). We propose an approach that bridges the gap between the term extraction algorithms that focus mainly on harvesting but do not taxonomize, and those that accept a new term and seek to enrich an already existing taxonomy. Our aim is to perform both stages: to extract the terms of a given domain and to induce their taxonomic organization without any initial taxonomic structure and information. This task is challenging because it is not trivial to discover both the hierarchically related and the parallel (perspectival) organizations of concepts. Achieving this goal can provide the research community with the ability to produce taxonomies for domains for which currently there are no existing or manually created ontologies.

3 Building Taxonomies from Scratch

3.1 Problem Formulation

We define our task as:

Task Definition: Given a root concept, a basic level concept or an instance, and recursive lexico-syntactic patterns, (1) harvest in bootstrapping fashion hyponyms and hypernyms subordinated to the root; (2) filter out erroneous information (extracted concepts and *isa* relations); (3) organize the harvested concepts into a taxonomy structure.

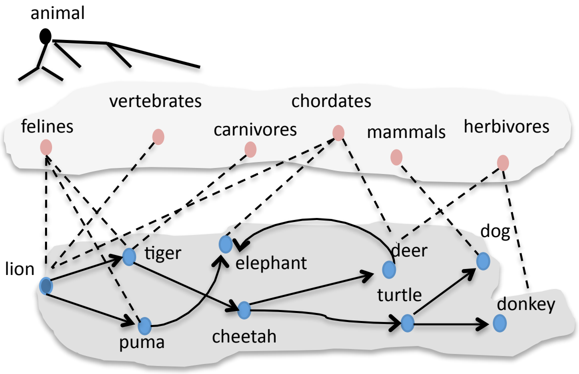


Figure 1: *Taxonomy Induction from Scratch.*

Figure 1 shows an example of the task. Starting with the root concept *animal* and the basic level concept *lion*, the algorithm learns new terms like *tiger, puma, deer, donkey* of class *animal*. Next for each basic level concept, the algorithm harvests hypernyms and learns that a *lion* is-a *vertebrate*, *chordate*, *feline* and *mammal*. Finally, the taxonomic structure of each basic level concept and its hypernyms is induced: *animal→chordate→vertebrate→mammal→feline→lion*.

3.2 Knowledge Harvesting

The main objective of our work is not the creation of a new harvesting algorithm, but rather the organization of the harvested information in a taxonomy structure starting from scratch. There are many algorithms for hyponym and hypernym harvesting from the Web. In our experiments, we use the doubly-anchored lexico-syntactic patterns and bootstrapping algorithm introduced by (Kozareva et al., 2008) and (Hovy et al., 2009).

We are interested in using this approach, because it is: (1) simple and easy to implement; (2) requires minimal supervision using only one root concept and a term to learn new hyponyms and hypernyms associated to the root; (3) reports higher precision than current semantic class algorithms (Etzioni et al., 2005; Pasca, 2004); and (4) adapts easily to different domains.

The general framework of the knowledge harvesting algorithm is shown in Figure 2.

```
 1. Given:
    a hyponym pattern $P_i$={$concept$ such as $seed$ and *}
    a hypernym pattern $P_c$={* such as $term_1$ and $term_2$}
    a root concept $root$
    a term called $seed$ for $P_i$
 2. build a query using $P_i$
 3. submit $P_i$ to Yahoo! or other search engine
 4. extract terms occupying the * position
 5. take terms from step 4 and go to step 2.
 6. repeat steps 2–5 until no new terms are found
 7. rank terms by outDegree
 8. for $\forall$ terms with outDegree>0, build a query
    using $P_c$
 9. submit $P_c$ to Yahoo! or other search engine
10. extract concepts (hypernyms) occupying the *
    position
11. rank concepts by inDegree
```

Figure 2: *Knowledge Harvesting Framework.*

The algorithm starts with a $root$ concept, *seed* term[1] of type *root* and a doubly-anchored pattern (DAP) such as '*<root> such as <seed> and **' which learns on the * position new terms of type *root*. The newly learned terms, which can be either instances, basic level or intermediate concepts, are placed into the position of the *seed* in the DAP pattern, and the bootstrapping process is repeated. The process ceases when no new terms are found.

To separate the true from incorrect terms, we use a graph-based algorithm in which each vertex u is a term, and an each edge $(u, v) \in E$ corresponds to the direction in which the term u discovered the term v. The graph is weighted $w(u, v)$ according to the number of times the term pair u-v is seen in unique web snippets. The terms are ranked by $outDegree(u)$= $\frac{\sum_{\forall(u,v)\in E} w(u,v)}{|V|-1}$ which counts the number of outgoing links of node u normalized by the total number of nodes in the graph excluding the current. The algorithm considers as true terms with $outDegree$>0.

All harvested terms are automatically fed into the hypernym extraction phase. We use the natural order in which the terms discovered each other and place them into an inverse doubly-anchored pattern (DAP^{-1}) '* *such as* <$term_1$> *and* <$term_2$>' to learn hypernyms on the * position. Similarly we build a graph with nodes h denoting the hypernyms and nodes t_1-t_2 denoting the term pairs. The edges $(h, t_1 - t_2) \in E'$ show the direction in which the term pair discovered the hypernym. The hypernyms are ranked by $inDegree(h)$= $\sum_{\forall(t_1-t_2,h)\in E'} w(t_1 - t_2, h)$ which rewards hypernyms that are frequently discovered by various term pairs. The output of the algorithm is a list of is-a relations between the learned terms (instances, basic level or intermediate concepts) and their corresponding hypernyms. For example, *deer is-a herbivore*, *deer is-a ruminant*, *deer is-a mammal*.

3.3 Graph-Based Taxonomy Induction

In the final stage of our algorithm, we induce the overall taxonomic structure using information about the pairwise positioning of the terms. In the knowledge harvesting and filtering phases, the algorithm learned is-a relations between the $root$ and the terms (instances, basic level or intermediate concepts), as well as the harvested hypernyms and the terms. The only missing information is the positioning of the intermediate concepts located between the basic level and the $root$ such as *mammals*, *vertibrates*, *felines*, *chordates*, among others.

We introduce a concept positioning (CP) procedure that uses a set of surface patterns: "X $such\ as$ Y", "X are Y $that$", "X $including$ Y", "X $like$ Y", "$such$ X as Y" to learn the hierarchical relations for all possible concept pairs. For each concept pair, say $chordates$ and $vertebrates$, we issue the two following queries:

(a) *chordates such as vertebrates*
(b) *vertebrates such as chordates*

[1] The input term can be an instance, a basic level or an intermediate concept. An intermediate concept is the one that is located between the basic level and root concepts.

If (a) returns more web hits than (b), then $chordates$ subsumes (or is broader than) $vertebrates$, otherwise $vertebrates$ subsumes $chordates$. For this pair the *such as* pattern returned 7 hits for (a) and 0 hits for (b), so that the overall magnitude of the direction of the relation is weak. To accumulate stronger evidence, we issue web queries with the remaining patterns. For the same concept pair, the overall magnitude of "X *including* Y" is 5820 hits for (a) and 0 for (b).

As shown in Figure 3, the concept positioning patterns cannot always determine the direct taxonomic organization between two concepts as in the case of *felines* and *chordates*, *felines* and *vertebrates*. One reason is that the concepts are located on distant taxonomic levels. We humans typically exemplify concepts using more proximate ones. Therefore, the concept positioning procedure can find evidence for the relation "*mammals→felines*", but not for "*chordates→felines*".

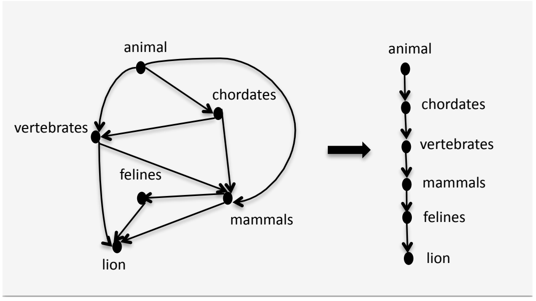


Figure 3: *Concept Positioning and Induced Taxonomy.*

After the concept positioning procedure has explored all concept pairs, we encounter two phenomena: (1) direct links between some concepts are missing and (2) multiple paths can be taken to reach from one concept to another.

To surmount these problems, we employ a graph based algorithm that finds the longest path in the graph G''=(V'', E''). The nodes V''=$\{it_1, h_1, h_2, .., h_n, r\}$ represent the input term, its hypernyms, and the $root$. An edge $(t_m, t_n) \in E''$ indicates that there is a path between the terms t_m and t_n. The direction $t_m \rightarrow t_n$ indicates the term subordination discovered during the CP procedure. The objective is to find the longest path in G'' between the $root$ and the input term. Intuitively, finding the longest paths is equivalent to finding the taxonomic organization of all concepts.

First, if present, we eliminate all cycles from the graph. Then, we find all nodes that have no predecessor and those that have no successor. Intuitively, a node with no predecessors p is likely to be positioned on the top of the taxonomy (e.g. $animal$), while a node with no successor s is likely to be located at the bottom (e.g. terms like *lion*, *tiger*, *puma*, or concepts like *krill predators* that could not be related to an instance or a basic level concept during the CP procedure). We represent the directed graph as an adjacency matrix $A = [a_{m,n}]$, where $a_{m,n}$ is 1 if (t_m, t_n) is an edge of G'', and 0 otherwise. For each (p, s) pair, we find the list of all paths connecting p with s. In the end, from all discovered candidate paths, the algorithm returns the longest one. The same graph-based taxonomization procedure is repeated for the rest of the basic level concepts and their hypernyms.

4 Experiments and Results

To evaluate the performance of a taxonomy induction algorithm, one can compare against a simple taxonomy composed of 2–3 levels. However, one cannot guarantee that the algorithm can learn larger hierarchies completely or correctly.

Animals provide a good example of the true complexity of concept organization: there are many types, they are of numerous kinds, people take numerous perspectives over them, and they are relatively well-known to human annotators. In addition, WordNet has a very rich and deep taxonomic structure for animals that can be used for direct comparison. We further evaluate our algorithm on the domains of *Plants* and *Vehicles*, which share some of these properties.

4.1 Data Collection

We have run the knowledge harvesting algorithm on the semantic classes *Animals*, *Plants* and *Vehicles* starting with only one seed example such as *lions*, *cucumbers* and *cars* respectively.

First, we formed and submitted the DAP pattern as web queries to Yahoo!Boss. We retrieved the top 1000 web snippets for each query. We kept all unique terms and term pairs. Second, we used the learned term pairs to form and submit new web queries DAP^{-1}. In this step, the algorithm harvested the hypernyms associated with each term. We kept all unique triples composed of a hypernym and the term pairs that extracted it. The algorithm ran until complete exhaustion for 8 iterations for *Animals*, 10 iterations for *Plants* and 18 iterations of *Vehicles*.

Table 1 shows the total number of terms extracted by the Web harvesting algorithm during the first stage. In addition, we show the number of terms that passed the $outDegree$ threshold. We found that the majority of the learned terms for *Animals* are basic level concepts, while for *Plants* and *Vehicles* they are a mixture of basic level and intermediate concepts.

	Animals	Plants	Vehicles
#Extracted Terms	1855	2801	1425
#outDegree(Term)$>$ 0	858	1262	581

Table 1: *Learned Terms.*

Since human based evaluation of all harvested terms is time consuming and costly, we have selected 90 terms located at the beginning, in the middle and in the end of the $outDegree$ ranking. Table 2 summarizes the results.

Plants	#CorrectByHand	#inWN	PrecByHand
rank[1-30]	29	28	.97
rank[420-450]	29	21	.97
rank[1232-1262]	27	19	.90
Vehicles	#CorrectByHand	#inWN	PrecByHand
rank[1-30]	29	27	.97
rank[193-223]	22	18	.73
rank[551-581]	25	19	.83

Table 2: *Term Evaluation.*

Independently, we can say that the precision of the harvesting algorithm is from 73 to 90%. In the case of *Vehicles*, we found that the learned terms in the middle ranking do not refer to the meaning of vehicle as a transportation devise, but to the meaning of vehicle as media (i.e. *seminar*, *newspapers*), communication and marketing. For the same category, the algorithm learned many terms which are missing from WordNet such as *BMW*, *bakkies*, *two-wheeler*, *all-terrain-vehicle* among others.

The second stage of the harvesting algorithm concerns hypernym extraction. Table 3 shows the total number of hypernyms harvested for all term pairs. The top 20 highly ranked concepts by $inDegree$ are the most descriptive terms for the domain. However, if we are interested in learning a larger set of hypernyms, we found that $inDegree$ is not sufficient by itself. For example, highly frequent but irrelevant hypernyms such as *meats*, *others* are ranked at the top of the list, while low frequent but relevant ones such as *protochordates*, *hooved-mammals*, *homeotherms* are discarded. This shows that we need to develop additional and more sensitive measures for hypernym ranking.

	Animals	Plants	Vehicles
#Extracted Hypernyms	1904	8947	2554
#inDegree(Hypernyms)$>$ 10	110	294	100

Table 3: *Learned Hypernyms.*

Table 4 shows some examples of the learned animal hypernyms which were annotated by humans as: correct but not present in WordNet; borderline which depending on the application could be valuable to have or exclude; and incorrect.

CorrectNotInWN	{colony|social} insects, grazers, monogastrics camelid, {mammalian|land|areal} predators {australian|african} wildlife, filter feeders hard shelled invertebrates, pelagics bottom dwellers
Borderline	prehistoric animals, large herbivores pocket pets, farm raised fish, roaring cats endangered mammals, mysterious hunters top predators, modern-snakes, heavy game
Incorrect	frozen foods, native mammals, red meats furry predators, others, resources, sorts products, items, protein

Table 4: *Examples of Learned Animal Hypernyms.*

The annotators found that 9% of the harvested is-a relations are missing from WordNet. For example, *cartilaginous_fish* → *shark*; *colony_insects*→ *bees*; *filter_feeders* → *tube_anemones* among others. This shows that despite its completeness, WordNet has still room for improvement.

4.2 A Test: Reconstructing WordNet

As previously discussed in (Hovy et al., 2009), it is extremely difficult even for expert to manually construct and evaluate the correctness of the harvested taxonomies. Therefore, we decided to evaluate the performance of our taxonomization approach reconstructing WordNet *Animals*, *Plants* and *Vehicles* taxonomies.

Given a domain, we select from 140 to 170 of the harvested terms. For each term, we retrieve all WordNet hypernyms located on the path between the input term and the $root$ that is *animal*, *plant* or *vehicle* depending on the domain of interest. We have found that 98% of the WordNet terms are also harvested by our knowledge acquisition algorithm. This means that being able to reconstruct WordNet's taxonomy is equivalent to evaluating the performance of our taxonomy induction approach.

Table 5 summarizes the characteristics of the taxonomies for the regions tested. For each domain, we show the total number of terms that must be organized, and the total number of is-a relations that must be induced.

	Animals	Plants	Vehicles
#terms	684	554	140
#is-a	4327	2294	412
average depth	6.23	4.12	3.91
max depth	12	8	7
min depth	1	1	1

Table 5: *Data for WordNet reconstruction.*

Among the three domains we have tested, *Animals* is the most complex and richest one. The maximum number of levels our algorithm must infer is 11, the minimum is 1 and the average taxonomic depth is 6.2. In total there are three basic level concepts (*longhorns*, *gaur* and *bullock*) with maximum depth, twenty terms (basic level and intermediate concepts) with minimum depth and ninety-eight terms (*wombat*, *viper*, *rat*, *limpkin*) with depth 6.

Plants is also a very challenging domain, because it contains a mixture of scientific and general terms such as *magnoliopsida* and *flowering plant*.

4.3 Evaluation

To evaluate the performance of our taxonomy induction approach, we use the following measures:

$$Precision = \frac{\#is{-}a\ found\ in\ WordNet\ and\ by\ system}{\#is{-}a\ found\ by\ system}$$

$$Recall = \frac{\#is{-}a\ found\ in\ WordNet\ and\ by\ system}{\#is{-}a\ found\ in\ WordNet}$$

Table 6 shows results of the taxonomy induction of the *Vehicles* domain using different concept positioning patterns. The most productive ones are: "X *are* Y *that*" and "X *including* Y". However, the highest yield is obtained when we combine evidence from all patterns.

Vehicles	Precision	Recall
X such as Y	.99 (174/175)	.42 (174/410)
X are Y that	.99 (206/208)	.50 (206/410)
X including Y	.96 (165/171)	.40 (165/410)
X like Y	.96 (137/142)	.33 (137/410)
such X as Y	.98 (44/45)	.11 (44/410)
AllPatterns	.99 (246/249)	**.60** (246/410)

Table 6: *Evaluation of the Induced Vehicle Taxonomy.*

Table 7 shows results of the taxonomization of the *Animals* and *Plants* domains. Overall, the obtained results are very encouraging given the fact that we started from scratch without the usage of any taxonomic structure. Precision is robust, but we must further improve recall. Our observation for the lower recall is that some intermediate concepts relate mostly to the high level ones, but not to the basic level concepts.

	Precision	Recall
Animals	.98 (1643/1688)	.38 (1643/4327)
Plants	.97 (905/931)	.39 (905/2294)

Table 7: *Evaluation of the Induced Animal and Plant Taxonomies.*

Figure 4 shows an example of the taxonomy induced by our algorithm for the *vipers*, *rats*, *wombats*, *ducks*, *emus*, *moths* and *penguins* basic level concepts and their WordNet hypernyms.

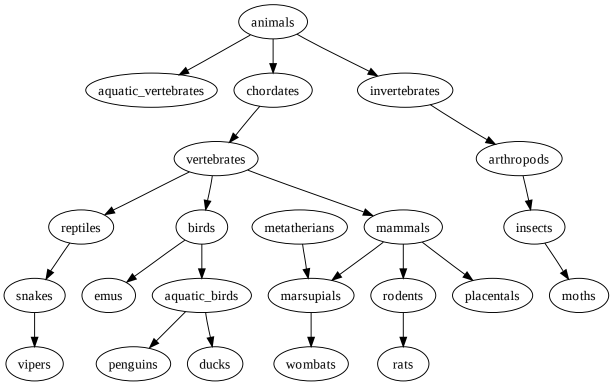


Figure 4: *Induced Taxonomy for Animals.*

The biggest challenge of the taxonomization process is the merging of independent taxonomic perspectives (a *deer* is a *grazer* in BehaviorByFeeding, a *wildlife* in BehaviorByHabitat, a *herd* in BehaviorSocialGroup and an *even-toed ungulate* in MorphologicalType) into a single hierarchy.

5 Conclusions and Future Work

We are encouraged by the ability of the taxonomization algorithm to reconstruct WordNet's *Animal* hierarchy, which is one of its most complete and elaborated. In addition, we have also evaluated the performance of our algorithm with the *Plant* and *Vehicle* WordNet hierarchies.

Currently, our automated taxonomization algorithm is able to build some of the quasi-independent perspectival taxonomies (Hovy et al., 2009). However, further research is required to develop methods that reliably (a) identify the number of independent perspectives a concept can take (or seems to take in the domain text), and (b) classify any harvested term into one or more of them. The result would greatly simplify the task of the taxonomization stage.

We note that despite this richness, WordNet has many concepts like *camelid*, *filter feeder*, *monogastrics* among others which are missing, but the harvesting algorithm can provide. Another promising line of research would investigate the combination of the two styles of taxonomization algorithms: first, the one described here to produce an initial (set of) taxonomies, and second, the term-insertion algorithms developed in prior work.

Acknowledgments

We acknowledge the support of DARPA contract number FA8750-09-C-3705. We thank Mark Johnson for the valuable discussions on taxonomy evaluation. We thank the reviewers for their useful feedback and suggestions.

References

Dmitry Davidov and Ari Rappoport. 2006. Efficient unsupervised discovery of word categories using symmetric patterns and high frequency words. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 297–304.

Dmitry Davidov, Ari Rappoport, and Moshel Koppel. 2007. Fully unsupervised discovery of concept-specific relationships by web mining. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 232–239.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: an experimental study. *Artificial Intelligence*, 165(1):91–134, June.

Roxana Girju, Adriana Badulescu, and Dan Moldovan. 2003. Learning semantic constraints for the automatic discovery of part-whole relations. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 1–8.

Oren Glickman, Ido Dagan, and Moshe Koppel. 2005. A probabilistic classification approach for lexical textual entailment. In *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference*, pages 1050–1055.

Marti Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics*, pages 539–545.

Eduard H. Hovy, Zornitsa Kozareva, and Ellen Riloff. 2009. Toward completeness in concept extraction and classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009*, pages 948–957.

Eduard Hovy. 2002. Comparing sets of semantic relations in ontologies. *The Semantics of Relationships: An Interdisciplinary Perspective*, pages 91–110.

Boris Katz and Jimmy Lin. 2003. Selectively using relations to improve precision in question answering. In *In Proceedings of the EACL-2003 Workshop on Natural Language Processing for Question Answering*, pages 43–50.

Zornitsa Kozareva, Ellen Riloff, and Eduard Hovy. 2008. Semantic class learning from the web with hyponym pattern linkage graphs. In *Proceedings of ACL-08: HLT*, pages 1048–1056.

Dekang Lin and Patrick Pantel. 2002. Concept discovery from text. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational linguistics*, pages 768–774.

Dan I. Moldovan, Sanda M. Harabagiu, Marius Pasca, Rada Mihalcea, Richard Goodrum, Roxana Girju, and Vasile Rus. 1999. Lasso: A tool for surfing the answer net. In *TREC*.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of 21st International Conference on Computational Linguistics and*

44th Annual Meeting of the Association for Computational Linguistics, ACL 2006.

Marius Pasca. 2004. Acquisition of categorized named entities for web search. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 137–145.

Marco Pennacchiotti and Patrick Pantel. 2006. Ontologizing semantic relations. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 793–800.

Ellen Riloff and Jessica Shepherd. 1997. A Corpus-Based Approach for Building Semantic Lexicons. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 117–124.

Alan Ritter, Stephen Soderland, and Oren Etzioni. 2009. What is this, anyway: Automatic hypernym discovery. In *Proceedings of AAAI Spring Symposium on Learning by Reading and Learning to Read*.

Alan Ritter, Mausam, and Oren Etzioni. 2010. A latent dirichlet allocation method for selectional preferences. In *to appear in Proceedings of the Association for Computational Linguistics ACL2010*.

Eleanor Rosch. 1978. Principles of categorization.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, ACL*.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 697–706.

Idan Szpektor, Ido Dagan, Roy Bar-Haim, and Jacob Goldberger. 2008. Contextual preferences. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 683–691.

Dominic Widdows. 2003. Unsupervised methods for developing taxonomies by combining syntactic and statistical information. In *Proceedings of HLT-NAACL*.

Hui Yang and Jamie Callan. 2009. A metric-based framework for automatic taxonomy induction. In *ACL-IJCNLP '09: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1*, pages 271–279.

Function-based question classification for general QA

Fan Bu, Xingwei Zhu, Yu Hao and **Xiaoyan Zhu**
State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Sci. and Tech., Tsinghua University
buf08@mails.tsinghua.edu.cn
etzhu192@hotmail.com
haoyu@mail.tsinghua.edu.cn
zxy-dcs@tsinghua.edu.cn

Abstract

In contrast with the booming increase of internet data, state-of-art QA (question answering) systems, otherwise, concerned data from specific domains or resources such as search engine snippets, online forums and Wikipedia in a somewhat isolated way. Users may welcome a more general QA system for its capability to answer questions of various sources, integrated from existed specialized sub-QA engines. In this framework, question classification is the primary task.

However, the current paradigms of question classification were focused on some specified type of questions, i.e. factoid questions, which are inappropriate for the general QA. In this paper, we propose a new question classification paradigm, which includes a question taxonomy suitable to the general QA and a question classifier based on MLN (Markov logic network), where rule-based methods and statistical methods are unified into a single framework in a fuzzy discriminative learning approach. Experiments show that our method outperforms traditional question classification approaches.

1 Introduction

During a long period of time, researches on question answering are mainly focused on finding short and concise answers from plain text for factoid questions driven by annual trackes such as CLEF, TREC and NTCIR. However, people usually ask more complex questions in real world which cannot be handled by these QA systems tailored to factoid questions.

During recent years, social collaborative applications begin to flourish, such as Wikipedia, Facebook, Yahoo! Answers and etc. A large amount of semi-structured data, which has been accumulated from these services, becomes new sources for question answering. Previous researches show that different sources are suitable for answering different questions. For example, the answers for factoid questions can be extracted from webpages with high accuracy, definition questions can be answered by corresponding articles in wikipedia(Ye et al., 2009) while community question answering services provide comprehensive answers for complex questions(Jeon et al., 2005). It will greatly enhance the overall performance if we can classify questions into several types, distribute each type of questions to suitable sources and trigger corresponding strategy to summarize returned answers.

Question classification (QC) in factoid QA is to provide constraints on answer types that allows further processing to pinpoint and verify the answer (Li and Roth, 2004). Usually, questions are classified into a fine grained content-based taxonomy(e.g. UIUC taxonomy (Li and Roth, 2002)). We cannot use these taxonomies directly. To guide question distribution and answer summarization, questions are classified according to their functions instead of contents.

Motivated by related work on user goal classification(Broder, 2002; Rose and Levinson, 2004) , we propose a function-based question classification category tailored to general QA. The category contain six types, namely **Fact**, **List**, **Reason**, **Solution**, **Definition** and **Navigation**. We will introduced this

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1119–1128,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

category in detail in Section 2.

To classify questions effectively, we unify rule-based methods and statistical methods into a single framework. Each question is splited into functional words and content words. We generate strict patterns from functional words and soft patterns from content words. Each strict pattern is a regular expression while each soft pattern is a bi-gram cluster. Given a question, we will evaluate its matching degree to each patterns. The matching degree is either 0 or 1 for strict pattern and between 0 and 1 for soft pattern. Finally, Markov logic network (MLN) (Richardson and Domingos, 2006) is used to combine and evaluate all the patterns.

The classical MLN maximize the probability of an assignment of truth values by evaluating the weights of each formula. However, the real world is full of uncertainty and is unnatural to be represented by a set of boolean values. In this paper, we propose fuzzy discriminative weight learning of Markov logic network. This method takes degrees of confidence of each evidence predicates into account thus can model the matching degrees between questions and soft patterns.

The remainder of this paper is organized as follows: In the next section we review related work on question classification, query classification and Markov logic network. Section 2 gives a detailed introduction to our new taxonomy for general QA. Section 4 introduces fuzzy discriminative weight learning of MLN and our methodology to extract strict and soft patterns. In Section 5 we compare our method with previous methods on Chinese question data from Baidu Zhidao and Sina iAsk. In the last section we conclude this work.

Although we build patterns and do experiments on Chinese questions, our method does not take advantage of the particularity of Chinese language and thus can be easily implemented on other languages.

2 Related Work

Many question taxonomies have been proposed in QA community. Lehnert (1977) developed the system QUALM based on thirteen conceptual categories which are based on a theory of memory representation. On the contrary, the taxonomy proposed by Graesser et al. (1992) has foundations both in theory and in empirical research. Both of these taxonomies are for open-domain question answering.

With the booming of internet, researches on question answering are becoming more practical. Most taxonomies proposed are focused on factoid questions, such as UIUC taxonomy (Li and Roth, 2002). UIUC taxonomy contains 6 coarse classes (**Abbreviation**, **Entity**, **Description**, **Human**, **Location** and **Numeric Value**) and 50 fine classes. All coarse classes are factoid oriented except **Description**. To classify questions effectively, Researchers have proposed features of different levels, such as lexical features, syntactic features (Nguyen et al., 2007; Moschitti et al., 2007) and semantic features (Moschitti et al., 2007; Li and Roth, 2004). Zhang and Lee (2003) compared five machine learning methods and found SVM outperformed the others.

In information retrieval community, researchers have described frameworks for understanding goals of user searches. Generally, web queries are classified into four types: **Navigational**, **Informational**, **Transactional** (Broder, 2002) and **Resource** (Rose and Levinson, 2004). Lee et al. (2005) automatically classify **Navigational** and **Informational** queries based on past user-click behavior and anchor-link distribution. Jansen and Booth (2010) investigate the correspondence between three user intents and eighteen topics. The result shows that user intents distributed unevenly among different topics.

Inspired by Rose and Levinson (2004)'s work in user goals classification, Liu et al. (2008) describe a three-layers cQA oriented question taxonomy and use it to determine the expected best answer types and summarize answers. Other than **Navigational**, **Informational** and **Transactional**, the first layer contains a new **Social** category which represents the questions that do not intend to get an answer but to elicit interaction with other people. **Informational** contains two subcategories **Constant** and **Dynamic**. **Dynamic** is further divided into **Opinion**, **Context-Dependent** and **Open**.

Markov logic network (MLN) (Richardson and Domingos, 2006) is a general model combining first-order logic and probabilistic graphical models in a single representation. Illustratively, MLN is a first-order knowledge base with a weight attached to each formula. The weights can be learnt ei-

TYPE	DESCRIPTION	EXAMPLES
1. Fact	People ask these questions for general facts. The expected answer will be a short phrase.	`Who is the president of United States?`

2. List	People ask these questions for a list of answers. Each answer will be a single phrase or a phrase with explanations or comments.	`List Nobel price winners in 1990s.`
		`Which movie star do you like best?`

3. Reason	People ask these questions for opinions or explanations. A good answer summary should contain a variety of opinions or comprehensive explanations. Sentence-level summarization can be employed.	`Is it good to drink milk while fasting?`
		`What do you think of Avatar?`

4. Solution	People ask these questions for problem shooting. The sentences in an answer usually have logical order thus the summary task cannot be performed on sentence level.	`What should I do during an earthquake?`
		`How to make pizzas?`

5. Definition	People ask these questions for description of concepts. Usually these information can be found in Wikipedia. If the answer is a too long, we should summarize it into a shorter one.	`Who is Lady Gaga?`
		`What does the Matrix tell about?`

6. Navigation	People ask these questions for finding websites or resources. Sometimes the websites are given by name and the resources are given directly.	`Where can I download the beta version of StarCraft 2?`

Table 1: Question Taxonomy for general QA

ther generatively (Richardson and Domingos, 2006) or discriminatively (Singla and Domingos, 2005). Huynh and Mooney (2008) applies -norm regularized MLE to select candidate formulas generated by a first-order logic induction system and prevent overfitting. MLN has been introduced to NLP and IE tasks such as semantic parsing (Poon et al., 2009) and entity relation extraction (Zhu et al., 2009).

3 A Question Taxonomy

We suggest a function-based taxonomy tailored to general QA systems by two principles. First, questions can be distributed into suitable QA subsystems according to their types. Second, we can employ suitable answer summarization strategy for each question type. The taxonomy is shown in Tab. 1.

At first glance, classifying questions onto this taxonomy seems a solved problem for English questions because of interrogative words. In most cases, a question starting with “Why” is for reason and “How” is for solution. But it is not always the case for other languages. From table 2 we can see two questions in Chinese share same function word “怎么样” but have different types.

In fact, even in English, only using interrogative words is not enough for function-based question classification. Sometimes the question content is crucial. For example, for question “Who is the current president of U.S. ?”, the answer is “Barak Obama” and the type is **Fact**. But for question “Who is Barak Obama?”, it will be better if we return the first paragraph from the corresponding Wiki article instead of a short phrase “current president of U.S.”. Therefore the question type will be **Definition**.

Compared to Wendy Lehnert’s or Arthur Graesser’s taxonomy, our taxonomy is more practical on providing useful information for question

Question	怎么样做宫保鸡丁？ How to cook Kung Pao Chicken?
Type	**Solution**
Question	大家觉得阿凡达怎么样？ What do you think of Avatar?
Type	**Reason**

Table 2: Two Chinese questions share same function words but have different types

extraction and summarization. Compared to ours, The UIUC taxonomy is too much focused on factoid questions. Apart from **Description**, all coarse types in UIUC can be mapped into **Fact**. The cQA taxonomy proposed in Liu et al. (2008) has similar goal with ours. But it is hard to automatically classify questions into that taxonomy, especially for types **Constant**, **Dynamic** and **Social**. Actually the author did not give implementation in the paper as well. To examine reasonableness of our taxonomy, we select and manually annotate 5800 frequent asked questions from Baidu Zhidao (see Section 5.1). The distribution of six types is shown in Fig. 1. 98.5 percent of questions can be categorized into our taxonomy. The proportion of each type is between 7.5% and 23.8%.

The type **Navigation** was originally proposed in IR community and did not cause too much concerns in previous QA researches. But from Fig. 1 we can see that navigational questions take a substantial proportion in cQA data.

Moreover, we can further develop subtypes for each type. For example, most categories in UIUC taxonomy can be regarded as refinement to **Fact** and **Navigation** can be refined into **Resource** and **Website**. We will not have further discussion on this issue.

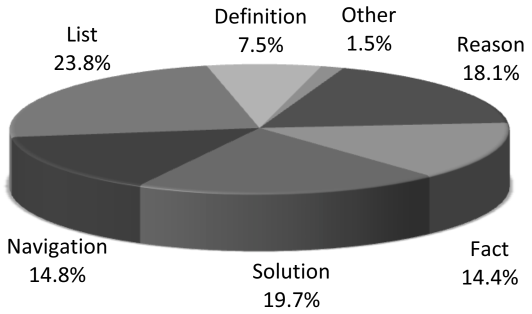


Figure 1: Distribution of six types in Baidu Zhidao data

4 Methodology

Many efforts have been made to take advantage of grammatical , semantic and lexical features in question classification. Zhang and Lee (2003) proposed a SVM based system which used tree kernel to incorporate syntactic features.

In this section, we propose a new question classification methodology which combines rule-based methods and statistical methods by Markov logic network. We do not use semantic and syntactic features for two reasons. First, the questions posted on online communities are casually written which cannot be accurately parsed by NLP tools, especially for Chinese. Second, the semantic and syntactic parsing are time consuming thus unpractical to be used in real systems.

We will briefly introduce MLN and fuzzy discriminative learning in section 4.1. The construction of strict patterns and soft patterns will be shown in 4.2 and 4.3. In section 4.4 we will give details on MLN construction, inference and learning.

4.1 Markov Logic Network

A first-order knowledge base contains a set of formulas constructed from logic operators and symbols for predicates, constants, variables and functions. An *atomic formula* or *atom* is a predicate symbol. Formulas are recursively constructed from atomic formulas using logical operators. The *grounding* of a predicate (formula) is a replacement of all of its arguments (variables) by constants. A *possible world* is an assignment of truth values to all possible groundings of all predicates.

In first-order KB, if a possible world violates even one formula, it has zero probability. Markov logic is a probabilistic extension and softens the hard constraints by assigning a weight to each formula. When a possible world violates one formula in the KB, it is less probable. The higher the weight, the greater the difference in log probability between a world that satisfies the formula and a world does not. Formally, Markov logic network is defined as follows:

Definition 1 *(Richardson & Domingos 2004) A Markov logic network L is a set of pairs (,), where is a formula in first-order logic and is a real number. Together with a finite set of constants C = , it defines a Markov network as follows:*

1. *contains one binary node for each possible grounding of each predicate appearing in L. The value of the node is 1 if the ground predicate is true, and 0 otherwise.*
2. *contains one feature for each possible grounding of each formula in L. The value of this feature is 1 if the ground formula is true, and 0 otherwise. The weight of the feature is the associated with in L.*

There is an edge between two nodes of iff the corresponding grounding predicates appear together in at least one grounding of one formula in . An MLN can be regarded as a template for constructing Markov networks. From Definition 1 and the definition of Markov networks, the probability distribution over possible worlds specified by the ground Markov network is given by

MLN weights can be learnt generatively(Richardson and Domingos, 2006) or discriminatively(Singla and Domingos, 2005). In discriminative weight learning, ground atom set is partitioned into a set of evidence atoms and a set of query atoms . The goal is to correctly predict the latter given the former. In this paper, we propose fuzzy discriminative weight learning which can take the prior confidence of each evidence atom into account.

Formally, we denote the ground formula set by . Suppose each evidence atom is given with a prior confidence , we define a confidence function as follows. For each ground atom , if then we have , else . For each ground non-atomic formulas, is defined on standard fuzzy operators, which are

We redefined the *conditional likelihood* of given as

Where is the set of ground formulas involving query atoms, is the set of formulas with at least one grounding involving a query atom and is the sum of confidence of the groundings of the ith formula involving query atoms. The gradient of the conditional log-likelihood (CLL) is

(1)

By fuzzy discriminative learning we can incorporate evidences of different confidence levels into one learning framework. Fuzzy discriminative learning will reduce to traditional discriminative learning when all prior confidences equal to 1.

4.2 Strict Patterns

In our question classification task, we find function words are much more discriminative and less sparse than content words. Therefore, we extract strict patterns from function words and soft patterns from content words. The definition of content and function words may vary with languages. In this paper, nouns, verbs, adjectives, adverbs, numerals and pronouns are regarded as content words and the rest are function words.

The outline of strict pattern extraction is shown in Alg. 1. In line 3, we build template by removing punctuations and replacing each character in each content word by a single dot. In line 4, we generate patterns from the template as follows. First we generate n-grams(n is between 2 and) from

```
Algorithm 1: Strict Pattern Extraction
  Input: Question Set ,
         Parameters and
  Output: Pattern Set
1 Initialize Pattern Set ;
2 for each Question do
3     String =ReplaceContentWords( , );
4     Pattern Set =GeneratePatterns( , );
5     for each Pattern in do
6         if in then
7             UpdateTypeFreq( , );
8         else
9             Add to ;
10 Merge similar patterns in ;
11 Sort by Information Gain on type
   frequencies;
12 return top Patterns in ;
```

during which each dot is treated as a character of zero length. For coverage concern, if a generated n-gram is not start(end) with dot, we build another n-gram by adding a dot before(behind) and add both and into n-gram set. Then for each n-gram, we replace each consecutive dot sequence by '.*' and the n-gram is transformed into a regular expression. A example is shown in Tab. 3. Although generated without exhaustively enumerating all possible word combinations, these regular expressions can capture most long range dependencies between function words.

Each pattern consists of a regular expression as well as its frequency in each type of questions. Still from Alg. 1, in line 5-9, if a pattern in question with type is found in , we just update the frequency of in , else is added to with only freq. equals to 1. In line 10, we merge similar patterns in . two patterns and are similar iff q Q`matchP` q p `matchP` q p , in which `matchP` is defined in Section 4.4.

Question	在网上可以开通网银吗？ Can I launch online banking services on internet?	
Template	在..可以....吗	
Patterns (=4)	.*以.*吗 .*可以.* .*可以.*吗.* .*在.*可以.*吗.* 在.*可以.* .*在.*可.*	.*以.*吗.* .*可以.*吗 .*在.*可以.* 在.*可.* 在.*可以.*吗

Table 3: Strict patterns generated from a question

Since a large number of patterns are generated, it is unpractical to evaluate all of them by Markov logic network. We sort patterns by information gain and only choose top "good" patterns in line 11-12 of Alg. 1. A "good" pattern should be discriminative and of wide coverage. The information gain IG of a pattern is defined as

IG

− − −

in which is the number of question types, is the probability of a question having type , (or −) is the probability of a question matching(or not matching) pattern . (or −) is the probability of a question having type given the condition that the question matches(or does not match) pattern . These probabilities can be approximately calculated by type and pattern frequencies on training data. From the definition we can see that information gain is suitable for pattern selection. The more questions a pattern matches and the more unevenly the matched questions distribute among questions types, the higher IG will be.

4.3 Soft Patterns

Apart from function words, content words are also important in function-based question classification. Content words usually contain topic information which can be a good complement to function words. Previous research on query classification(Jansen and Booth, 2010) shows that user intents distribute unevenly among topics. Moreover, questions given by users may be incomplete and contain not function words. For these questions, we can only predict the question types from topic information.

Compared with function words, content words distribute much more sparsely among questions.

When we represent topic information by content words (or bi-grams), since the training set are small and less frequent words (or bi-grams) are filtered to prevent over-fitting, those features would be too sparse to predict further unseen questions.

To solve this problem, we build soft patterns on question set. Each question is represented by a weighted vector of content bi-grams in which the weight is bi-gram frequency. Cosine similarity is used to compute the similarity between vectors. Then we cluster question vectors using a simple single-pass clustering algorithm(Frakes and Yates, 1992). That is, for each question, we compute its similarity with each centroid of existing cluster. If the similarity with nearest cluster is greater than a minimum similarity threshold , we assign this question to that cluster, else a new cluster is created for this question.

Each cluster is defined as a soft pattern. Unlike strict patterns, a question can match a soft pattern to some extent. In this paper, the degree of matching is defined as the cosine similarity between question and centroid of cluster. Soft patterns are flexible and could alleviate the sparseness of content words. Also, soft patterns can be pre-filtered by information gain described in 4.2 if necessary.

4.4 Implementation

Currently, we model patterns into MLN as follows. The main query predicate is Type(q,t), which is true iff question q has type t. For strict patterns, the evidence predicate MatchP(q,p) is true iff question q is matched by strict pattern p. The confidence of MatchP(q,p) is 1 for each pair of (q,p). For soft patterns, the evidence predicate MatchC(q,c) is true iff the similarity of question q and the cluster c is greater than a minimum similarity requirement . If MatchC(q,c) is false, its confidence is 1, else is the similarity between q and c.

We represent the relationship between patterns and types by a group of formulas below.

MatchP(q,+p) Type(q,+t) Type(q,t')

The "+p, +t" notation signifies that the MLN contains an instance of this formula for each (*pattern*, *type*) pair. For the sake of efficacy, for each pattern-type pair (p,t), if the proportion of type t in questions matching p is less than a minimum requirement , we remove corresponding formula from MLN.

Similarly, we incorporate soft patterns by

MatchC(q,+c) Type(q,+t) Type(q,t')

Our weight learner use -regularization (Huynh and Mooney, 2008) to select formulas and prevent overfitting. A good property of -regularization is its tendency to force parameters to exact zero by strongly penalizing small terms (Lee et al., 2006). After training, we can simply remove the formulas with zero weights.

Formally, to learn weight for each formula, we iteratively solve -norm regularized optimization problem:

where is -norm and parameter controls the penalization of non-zero weights. We implement the Orthant-Wise Limited-memory Quasi-Newton algorithm(Andrew and Gao, 2007) to solve this optimization.

Since we do not model relations among questions, the derived markov network can be broken up into separated subgraphs by questions and the gradient of CLL(Eq. 1) can be computed locally on each subgraph as

(2)

in which and are the evidence and query atoms involving question . Eq. 2 can be computed fast without approximation.

We initialize formula weights to the same positive value . Iteration started from uniform prior can always converge to a better local maximum than gaussian prior in our task.

5 Experiments

5.1 Data Preparation

To the best of our knowledge, there is not general QA system(the system which can potentially answer all kinds of questions utilizing data from heterogeneous sources) released at present. Alteratively, we test our methodology on cQA data based on observation that questions on cQA services are of various length, domain independent and wrote informally(even with grammar mistakes). General QA systems will meet these challenges as well.

In our experiments, both training and test data are from Chinese cQA services Baidu Zhidao and Sina iAsk. To build training set, we randomly select 5800 frequent-asked questions from Baidu Zhidao. A question is frequent-asked if it is lexically similar to at least five other questions. Then we ask 10 native-speakers to annotate these questions according to question title and question description. If an annotator cannot judge type from question title, he can view the question description. If type can be judged from the description, the question title will be replaced by a sentence selected from it. If not, this question will be labeled as **Other**.

Each question is annotated by two people. If a question is labeled different types, another annotator will judge it and make final decision. If this annotator cannot judge the type, this question will also be labeled as **Other**. As a result, disagreements show up on eighteen percents of questions. After the third annotator's judgment, the distribution of each type is shown in Fig. 1.

To examine the generalization capabilities, the test data is composed of 700 questions randomly selected from Baidu Zhidao and 700 questions from Sina iAsk. The annotation process on test data is as same as the one on training data.

5.2 Methods Compared and Results

We compare four methods listed as follows.

SVM with bi-grams. We extract bi-grams from questions on training data as features. After filtering the ones appearing only once, we collect 5700 bi-grams. LIBSVM(Chang and Lin, 2001)is used as the multi-class SVM classifier. All parameters are adjusted to maximize the accuracy on test data. We denote this method as "*SB*";

MLN with bi-grams. To compare MLN and SVM, we treat bi-grams as strict patterns. If a question contain a bi-gram, it matches the corresponding pattern. We set , and . As a result, 5700 bi-grams are represented by 10485 formulas. We denote this method as "*MB*";

MLN with strict patterns and bi-grams. We ask two native-speakers to write strict patterns for each type. The pattern writers can view training data for reference and write any Java-style regular expressions. Then we carefully choose 50 most reliable patterns. To overcome the low coverage, We also use the method described in Sec. 4.2 to automatically extract strict patterns from training set. We first select top 3000 patterns by information gain, merge these patterns with hand-crafted ones and combine similar patterns. Then we represent these patterns by formulas and learn the weight of each formula by MLN. After removing the formula with low weights, we finally retain 2462 patterns represented by 3879 formulas. To incorporate content information, we extract bi-grams from questions with function words removed and remove the ones with frequency lower than two. With bi-grams added, we get 8173 formulas in total. All parameters here are the same as in "*MB*". We denote this method as "*MSB*";

MLN with strict patterns and soft patterns. To incorporate content information, We cluster questions on training data with similarity threshold

and get 2588 clusters(soft patterns) which are represented by 3491 formulas. We these soft patterns with strict patterns extracted in "*MSB*", which add up to 7370 formulas. We set and the other parameters as same as in "*MB*". We denote this method as "*MSS*";

We separate test set into easy set and difficult set. A question is classified into easy set iff it contains function-words. As a result, the easy set contains 1253 questions. We measure the accuracy of these four methods on easy data and the whole test data. The results are shown in Tab 4. From the results we can see that all methods perform better on easy questions and MLN outperforms SVM using same bi-gram features. Although *MSS* is inferior to *MSB* on

	F. num	Easy data	All data
SB	NA	0.724	0.685
MB	10485	0.722	0.692
MSB	8173	**0.754**	0.714
MSS	7370	0.752	**0.717**

Table 4: Experimental results on Chinese cQA data

	F	L	S	R	D	N
Prec.	0.63	0.65	0.83	0.76	0.69	0.55
Recall	0.55	0.74	0.86	0.76	0.44	0.58
	0.59	0.69	0.84	0.76	0.54	0.56

Table 5: Precision, recall and F-score on each type

easy questions, it shows better overall performance and uses less formulas.

We further investigate the performance on each type. The precision, recall and -score of each type by method *MSS* are shown in Tab. 5. From the results we can see that the performance on **Solution** and **Reason** are significantly better than the others. It is because the strict patterns for this two types are simple and effective. A handful of patterns could cover a wide range of questions with high precision. It is difficult to distinguish **Fact** from **List** because strict patterns for these two types are partly overlap each other. Sometimes we need content information to determine whether the answer is unique. Since **List** appears more frequently than **Fact** on training set, MLN tend to misclassify **Fact** to **List** which lead to low recall of the former and low precision of the latter. The recall of **Definition** is very low because many definition questions on test set are short and only consists of content words(e.g. a noun phrase). This shortage could be remedied by building strict patterns on POStagging sequence.

fraction lines, college entrance exam
分数，数线，高考，考分，录取，... Fact: 56.4% List: 33.3% Solu.: 5.5%
lose weight, summer, fast
减肥，夏天，快速，速减，方法，... Reas.: 53.8% Solu.: 42.3% List: 3.8%
TV series, interesting, recent
电视，视剧，好看，最近，最新，... List: 84.0% Fact: 8.0% Navi.: 2.0%
converter, format, 3gp
转换，换器，3gp，mp4，格式，... Navi.: 75% List: 18.8% Solu.: 6.2%

Table 6: Selected soft patterns on training data

5.3 Case Study on Soft Patterns

To give an intuitive illustration of soft patterns, we show some of them clustered on training data in Tab. 6. For each soft pattern, we list five most frequent bi-grams and its distribution on each type(only top 3 frequent types are listed).

From the results we can see that soft patterns are consistent with our ordinary intuitions. For example, if user ask a questions about “TV series”, he is likely to ask for recommendation of recent TV series and the question have a great chance to be **List**. If user ask questions about “lose weight”, he probably ask something like “How can I lose weight fast?” or “Why my diet does not work?” . Thus the type is likely to be **Solution** or **Reason**.

6 Conclusion and Future Work

We have proposed a new question taxonomy tailored to general QA on heterogeneous sources. This taxonomy provide indispensable information for question distribution and answer summarization. We build strict patterns and soft patterns to represent the information in function words and content words. Also, fuzzy discriminative weight learning is proposed for unifying strict and soft patterns into Markov logic network.

Currently, we have not done anything fancy on the structure of MLN. We just showed that under uniform prior and L1 regularization, the performance of MLN is comparable to SVM. To give full play to the advantages of MLN, future work will focus on fast structure learning. Also, since questions on online communities are classified into categories by topic, we plan to perform joint question type inference on function-based taxonomy as well as topic-based taxonomy by Markov logic. The model will not only capture the relation between patterns and types but also the relation between types in different taxonomy.

Acknowledgment

This work was supported mainly by Canada’s IDRC Research Chair in Information Technology program, Project Number: 104519-006. It is also supported by the Chinese Natural Science Foundation grant No. 60973104.

References

G. Andrew and J. Gao. 2008. *Scalable training of L1-regularized log-linear models*. In Proc. of ICML 2007, pp. 33-40.

A. Broder. 2002. *A taxonomy of Web search*. SIGIR Forum, 36(2), 2002.

C.C. Chang and C.J. Lin. 2001. *LIBSVM: a library for support vector machines*. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

W.B. Frakes and R. Baeza-Yates, editors. 1992. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall, 1992.

A.C. Graesser, N.K. Person and J.D. Huber. 1992. *Mechanisms that generate questions*. Questions and Information Systems, pp. 167-187), Hillsdale, N.J.: Erlbaum.

T.N. Huynh and R.J. Mooney. 2008. *Discriminative Structure and Parameter Learning for Markov Logic Networks*. In Proc. of ICML 2008, pp. 416-423.

B.J. Jansen and D. Booth. 2010. *Classifying web queries by topic and user intent*. In Proc. of the 28th international conference on human factors in computing systems, pp. 4285-4290.

J. Jeon, W.B. Croft and J.H. Lee. 2005. *Finding similar questions in large question and answer archives*. In Proc. of ACM CIKM 2005,pp. 76-83.

S. Lee, V. Ganapathi and D. Koller. 2005. *Efficient structure learning of Markov networks using -regularization.*. Advances in Neural Information Processing Systems 18.

U. Lee, Z. Liu and J. Cho. 2005. *Automatic identification of user goals in Web search*. In Proc. of WWW 2005.

W. Lehnert. 1977. *Human and computational question answering*. Cognitive Science, vol. 1, 1977, pp. 47-63.

X. Li and D. Roth. 2002. *Learning question classifiers*. In Proc. of COLING 2002, pp. 556-562.

X. Li and D. Roth. 2004. *Learning question classifiers: the role of semantic information*. Natural Language Engineering.

Y. Liu, S. Li, Y. Cao, C.Y. Lin, D. Han and Y. Yu. 2008. *Understanding and summarizing answers in community-based question answering services*. In Proc. of COLING 2008.

A. Moschitti, S. Quarteroni, R. Basili and S. Manandhar. 2007. *Exploiting Syntactic and Shallow Semantic Kernels for Question/Answer Classification*. In Proc. of ACL 2007.

M.L. Nguyen, T.T. Nguyen and A. Shimazu. 2007. *Subtree Mining for Question Classification Problem*. In Proc. of IJCAI 2007.

H. Poon and P. Domingos. 2009. *Unsupervised semantic parsing*. In Proc. of EMNLP 2009, pp. 1-10

D.E. Rose and D. Levinson. 2004. *Understanding user goals in web search*. In Proc. of WWW 2004.

M. Richardson and P. Domingos. 2006. *Markov logic networks*. Machine Learning 62:107-136.

P. Singla and P. Domingos. 2005. *Discriminative Training of Markov Logic Networks*. In Proc. of AAAI 2005.

S. Ye, T.S. Chua and J. Lu. 2009. *Summarizing Definition from Wikipedia*. In Proc. of ACL 2009.

D. Zhang and W.S. Lee. 2003. *Question classification using support vector machines*. In Proc. of ACM SIGIR 2003, pp. 26-32.

J. Zhu , Z. Nie, X. Liu, B. Zhang and J.R. Wen. 2009. *StatSnowball: a Statistical Approach to Extracting Entity Relationships*. In Proc. of WWW 2009, pp. 101-110

Learning Recurrent Event Queries for Web Search

Ruiqiang Zhang and **Yuki Konda** and **Anlei Dong**
Pranam Kolari and **Yi Chang** and **ZhaohuiZheng**
Yahoo! Inc
701 First Avenue, Sunnyvale, CA94089

Abstract

Recurrent event queries (REQ) constitute a special class of search queries occurring at regular, predictable time intervals. The freshness of documents ranked for such queries is generally of critical importance. REQ forms a significant volume, as much as 6% of query traffic received by search engines. In this work, we develop an improved REQ classifier that could provide significant improvements in addressing this problem. We analyze REQ queries, and develop novel features from multiple sources, and evaluate them using machine learning techniques. From historical query logs, we develop features utilizing query frequency, click information, and user intent dynamics within a search session. We also develop temporal features by time series analysis from query frequency. Other generated features include word matching with recurrent event seed words and time sensitivity of search result set. We use Naive Bayes, SVM and decision tree based logistic regression model to train REQ classifier. The results on test data show that our models outperformed baseline approach significantly. Experiments on a commercial Web search engine also show significant gains in overall relevance, and thus overall user experience.

1 Introduction

REQ pertains to queries about events which occur at regular, predictable time intervals, most often weekly, monthly, annually, bi-annually, etc. Naturally, users issue REQ periodically. REQ usually refer to:

Organized public events such as festivals, conferences, expos, sports competitions, elections: winter olympics, boston marathon, the International Ocean Research Conference, oscar night.

Public holidays and other noteworthy dates: labor day, date of Good Friday, Thanksgiving, black friday.

Products with annual model releases, such as car models: ford explorer, prius.

Lottery drawings: California lotto results.

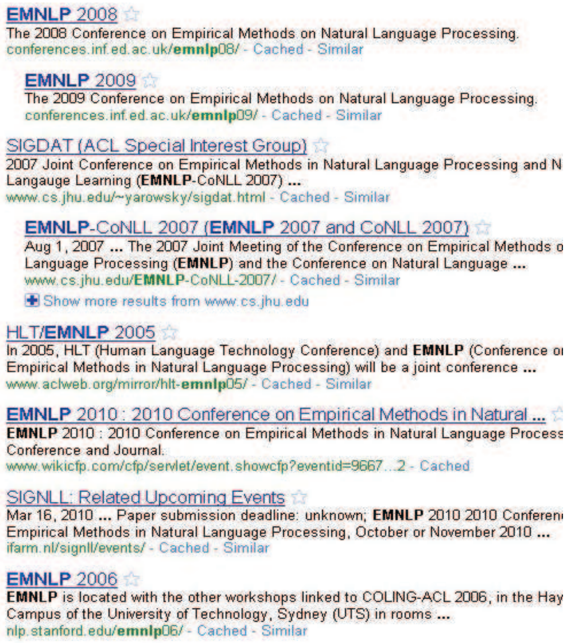
TV shows and programs which are currently running: American idol, Inside Edition.

Cultural related activities: presidential election, tax return, 1040 form.

Our interest in studying REQ arises from the challenge imposed on Web search ranking. To illustrate this, we show an example in Fig. 1 that snapshots the real ranking results of the query, *EMNLP*, issued in 2010 when the authors composed this paper, on Google search engine. It is obvious the ranking is not satisfactory because the page about *EMNLP2008* is on the first position in 2010. Ideally, the page about *EMNLP2010* on the 6th position should be on the first position even if users don't explicitly issue the query, *EMNLP 2010*, because *EMNLP* is a REQ. The query, "EMNLP", implicitly, without a year qualifier, needs to be served the most recent pages about "EMNLP".

A better search ranking result cannot be achieved if we do not categorize "EMNLP" as a REQ, and provide special ranking treatment to such queries. Existing search engines adopt a fairly involved ranking algorithm to order Web search results by considering many factors. Time is an important factor but not the most critical. The page's ranking score mostly depends on other features such as tf-idf (Salton and McGill, 1983), BM25 (Jones

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1129–1139,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics



EMNLP 2008
The 2008 Conference on Empirical Methods on Natural Language Processing.
conferences.inf.ed.ac.uk/**emnlp**08/ - Cached - Similar

EMNLP 2009
The 2009 Conference on Empirical Methods on Natural Language Processing.
conferences.inf.ed.ac.uk/**emnlp**09/ - Cached - Similar

SIGDAT (ACL Special Interest Group)
2007 Joint Conference on Empirical Methods in Natural Language Processing and N
Langauge Learning (**EMNLP**-CoNLL 2007) ...
www.cs.jhu.edu/~yarowsky/sigdat.html - Cached - Similar

EMNLP-CoNLL 2007 (**EMNLP** 2007 and CoNLL 2007)
Aug 1, 2007 ... The 2007 Joint Meeting of the Conference on Empirical Methods o
Language Processing (**EMNLP**) and the Conference on Natural Language ...
www.cs.jhu.edu/**EMNLP**-CoNLL-2007/ - Cached - Similar

Show more results from www.cs.jhu.edu

HLT/**EMNLP** 2005
In 2005, HLT (Human Language Technology Conference) and **EMNLP** (Conference o
Empirical Methods in Natural Language Processing) will be a joint conference ...
www.aclweb.org/mirror/hlt-**emnlp**05/ - Cached - Similar

EMNLP 2010 : 2010 Conference on Empirical Methods in Natural ...
EMNLP 2010 : 2010 Conference on Empirical Methods in Natural Language Process
Conference and Journal.
www.wikicfp.com/cfp/servlet/event.showcfp?eventid=9667...2 - Cached

SIGNLL: Related Upcoming Events
Mar 16, 2010 ... Paper submission deadline: unknown; **EMNLP** 2010 2010 Conferen
Empirical Methods in Natural Language Processing, October or November 2010 ...
ifarm.nl/signll/events/ - Cached - Similar

EMNLP 2006
EMNLP is located with the other workshops linked to COLING-ACL 2006, in the Hay
Campus of the University of Technology, Sydney (UTS) in rooms ...
nlp.stanford.edu/**emnlp**06/ - Cached - Similar

Figure 1: A real problematic ranking result by Google for a REQ query, “EMNLP”. The EMNLP2010 page should be on the 1st position.

et al., 2000), anchor text, historical clicks, pagerank (Brin and Page, 1998), and overall page quality. New pages about EMNLP2010 obtain less favorable feature values than the pages of 2009 earlier in terms of anchor text, click or pagerank because they have existed for a shorter time and haven’t accumulated sufficient popularity to make them stand out. Without special treatment, the new pages about “EMNLP2010” will typically not be ranked appropriately for the users.

Typically, a recurrent event is associated with a root, and spawns a large set of queries. Oscar, for instance, is a recurrent event about the annual Academy Award. Based on this, queries like “oscar best actress”, “oscar best dress”, “oscar best movie award”, are all recurrent event queries. As such, REQ is a highly frequent category of query in Web search. By Web search query log analysis, we observe that there about 5-6% queries of total query volume belongs to this category.

In this work, we learn if a query is in the REQ class, by effectively combining multiple features. Our features are developed through analysis of historical query logs. We discuss our approaches in detail in Section 3. We then develop a REQ classifier where all the features are integrated by machine learning models. We use Naive Bayes, SVM and decision tree based logistic regression models. These models are described in Section 4. Our experiments for REQ classifier and Web search ranking are detailed in Section 5 and 6.

2 Related Work

We found our work were related to two other problems: general query classification and time-sensitive query classification. For general query classification, the task is to assign a Web search query to one or more predefined categories based on its topics. In the query classification contest in KDD-CUP 2005 (Li et al., 2005), seven categories and 67 sub-categories were defined. The winning solution (Shen et al., 2005) used multiple classifiers integrated by ensemble method. The difficulties for query classification are from short queries, lack of labeled data, and query sense ambiguity. Most popular studies use query log, web search results, unlabeled data to enrich query classification (Shen et al., 2006; Beitzel et al., 2005), or use document classification to predict query classification (Broder et al.,). General query classification is also studied for query intent detection by (Li et al., 2008).

There are many prior works to study the time sensitivity issue in web search. For example, Baeza-Yates *et al.* (Baeza-Yates et al., 2002) studied the relation between the web dynamics, structure and page quality, and demonstrated that PageRank is biased against new pages. In T-Rank Light and T-Rank algorithms (Berberich et al., 2005), both activity (i.e., update rates) and freshness (i.e., timestamps of most recent updates) of pages and links are taken into account for link analysis. Cho *et al.* (Cho et al., 2005) proposed a page quality ranking function in order to alleviate the problem of popularity-based ranking, and they used the derivatives of PageRank to forecast future PageRank values for new pages. Pandey *et al.* (Pandey et al., 2005) studied the tradeoff between new page exploration and high-quality page exploitation, which is based on a ranking method to randomly promote some new pages so that they can accumulate links quickly.

More recently, Dong *et al.* (Dong et al., 2010a) proposed a machine-learned framework to improve ranking result freshness, in which novel features, modeling algorithms and editorial guideline are used to deal with time sensitivities of queries and documents. In another work (Dong et al., 2010b), they use micro-blogging data (e.g., Twitter data) to detect fresh URLs. Novel and effective features are also extracted for fresh URLs so that ranking recency in web search is improved.

Perhaps the most related work to this paper is the query classification approach used in (Zhang et al., 2009) and (Metzler et al., 2009), in which year qualified queries (YQQs) are detected based on heuristic rules. For example, a query containing a year stamp is an explicit YQQ; if the year stamp is removed from this YQQ, the remaining part of this query is also a YQQ, which is called implicit YQQ. Different ranking approaches were used in (Zhang et al., 2009) and (Metzler et al., 2009) where (Zhang et al., 2009) boosted pages of the most latest year while (Metzler et al., 2009) promoted pages of the most influential years. Similarly, Nunes *et al.* (Nunes, 2007) applied information extraction techniques to identify temporal expression in web search queries, and found 1.5% of queries containing temporal expression.

Dong *et al.* (Dong et al., 2010a) proposed a breaking-news query classifier with high accuracy and reasonable coverage, which works not by modeling each individual topic and tracking it over time, but by modeling each discrete time slot, and comparing the models representing different time slots. The buzziness of a query is computed as the language model likelihood difference between different time slots. In this approach, both query log and news contents are exploited to compute language model likelihood.

Diaz (Diaz, 2009) determined the newsworthiness of a query by predicting the probability of a user clicks on the news display of a query. In this framework, the data sources of both query log and news corpus are leveraged to compute contextual features. Furthermore, the online click feedback also plays a critical role for future click prediction.

Konig *et al.* (Knig et al., 2009) estimated the click-through rate for dedicated news search result with a supervised model, which is to satisfy the requirement of adapting quickly to emerging news event. Some additional corpora such as blog crawl and Wikipedia is used for buzziness inference. Compared with (Diaz, 2009), different feature and learning algorithms are used.

Elsas *et al.* (Elsas and Dumais, 2010) studied improving relevance ranking by detecting document content change to leverage temporal information.

3 Feature Generation

To better understand our work, we first introduce three terms. We subdivide all raw queries in query log into three categories: Explicit Timestamp, Implicit Timestamp, and No Timestamp. An Explicit Timestamp query contains at least one token being a time indicator. For example, *emnlp 2010*, *2007 December holiday calendar*, *amsterdam weather summer 2009*, *Google Q1 reports 2010*. These queries are considered to conatin time indicators, because we can regard {*2010, 2007, 2009*} as year indicator, *december* as *month* indicator, {*summer, Q1(first quarter)*} as *seasonal* indicator. To simplify our work, we only consider the *year* indicators, *2010, 2007, 2009*. Such *year* indicators are also the most important and most popular indicators, as noted in (Zhang et al., 2009). Any query containing at least one *year* indicator is an Explicit Timestamp query. Due to word sense ambiguity, some queries labeled as Explicit Timestamp by this method may have no connection with time such as *Windows Office 2007*, *2010 Sunset Boulevard*, or *call number 2008*. In this work, we tolerate this type of error because word sense disambiguation is a peripheral problem for this task.

Implicit Timestamp queries are resulted by removing all *year* indicators from the corresponding Explicit Timestamp queries. For example, the Implicit Timestamp query of *emnlp 2010* is *emnlp*. All other queries are No Timestamp queries because they have never been found together with a *year* indicator.

Classifying queries into the above three categories depends on the used query log. A search engine company partner provided us a query log from 08/01/2009 to 02/29/2010 for this research. We found the proportions of the three categories in this query log are 13.8% (Explicit), 17.1% (Implicit) and 69.1% (No Timestamp). These numbers could be slightly different depending on the source of query logs. Note that 17.1% of Implicit Timestamp queries in the query log is a significant number. However, not all Implicit Timestamp queries are REQ. Many Implicit Timestamp queries have no time sense. They belong to Implicit Timestamp just because users issued the query with a *year* indicator through varied intents. For example, “google” is found to be an Implicit Timestamp query since there were many “google 2008” or “google 2009” in the query log.

The next few sections introduce our work in recognizing recurrent event time sense for Implicit Timestamp queries. We first focus on features. There are many features that were exploited in REQ classifier. We extract these features from query log, query session log, click log, search results, time series and NLP morphological analysis.

3.1 Query log analysis

The following features are extracted from query log analysis:

QueryDailyFrequency: the total counts of the query divided by the number of the days in the period.

ExplicitQueryRatio: Ratio of number of counts query was issued with year and number of counts query was issued with or without year. This feature is the method used by (Zhang et al., 2009).

UniqExplicitQueryCount: Number of uniq Explicit Timestamp queries associated with query. For example, if a query was issued with query+2009 and query+2008, this feature’s value is two.

ChiSquareYearDist: this feature is the distance between two distributions: one is frequency distribution over years for all REQ queries. The other is that for single REQ query. It is calculated through following steps: (a) Aggregate the frequencies for all queries for all years. Suppose we observe all years from 2001 to 2010. So we can get vector, $E = (\frac{af_{10}}{sum1}, \frac{af_{09}}{sum1}, ..., \frac{af_{01}}{sum1})$ where af_i is the frequency sum of year 20i for all REQ queries. $sum1 = af_{10} + af_{09} + ... + af_{01}$, the sum of all year frequency. (b) Given a query, suppose we observe this query’s yearly frequency distribution is , $O^q = (qf_{10}, qf_{09}, , ..., qf_{01})$. qf_i is this query’s frequency for the year 20i. Pad the slot with zeros if no frequency found. The expected distribution for this query is, $E^q = (\frac{sum2*af_{10}}{sum1}, \frac{sum2*af_{09}}{sum1}, ..., \frac{sum2*af_{01}}{sum1})$, where $sum2 = qf_{10} + qf_{09} + ... + qf_{01}$ is sum of all year frequency for the query. (d) Calculate CHI-squared value to represent the different yearly frequency distribution between E^q and O^q according to $\chi^2 = \sum_{i=1}^{N} \frac{(O_i^q - E_i^q)^2}{E_i^q}$. Using CHI square distance as a method is widely used for statistical hypothesis test. We found it to be a useful feature for REQ classifier.

3.2 Query reformulation

If users cannot find the newest page by issuing Implicit Timestamp query, they may re-issue the query using an Explicit Timestamp query. We can detect this change in a search session (a 30 minutes period for each query). By finding this kind of behavior from users, we next extract three features.

UserSwitch: Number of unique users that switched from Implicit Timestamp queries to Explicit Timestamp queries.

YearSwitch: Number of unique year-like tokens switched by users in a query session.

NormalizedUserSwitch: Feature UserSwitch divided by QueryDailyFrequency.

3.3 Click log analysis

If a query is time sensitive, users may click a page that displays the year indicator on title or url. An example that shows year indicator on url is *www.lsi.upc.edu/events/emnlp2010/call.html*. Search engine click log saves all users’ click information. We used click log to derive the following features.

YearUrlTop5CTR: Aggregated click through rate (CTR) of all top five URLs containing a year indicator. CTR of an URL is defined as the number of clicks of an URL divided by the number of page views.

YearUrlFPCTR: Aggregated click through rate (CTR) of all first page URLs containing a year indicator.

3.4 Search engine result set

For each Implicited Timestamp query, we can scrape the search engine to get search results. We count the number of titles and urls that contain year indicator. We use this number as a feature, and generate 6 features.

TitleYearTop5: the number of titles containing a year indication on the top 5 results. This value is 4 in Fig. 1.

TitleYearTop10: the number of titles containing a year indication on the top 10 results. This value is 6 in Fig. 1.

TitleYearTop30: the number of titles containing a year indication on the top 30 results.

UrlYearTop5: the number of urls containing a year indication on the top 5 results. This value is 1 in Fig. 1.

UrlYearTop10: the number of urls containing a year indication on the top 10 results.

UrlYearTop30: the number of titles containing a year indication on the top 30 results.

3.5 Time series analysis

Recurrent event query has periodic occurrence pattern in time series. Top graph of Figure 2 shows the frequency change of the query, “Oscar”. The annual event usually starts from Oscar nomination as earlier as last year December to award announcement of February this year. So a small spike and a big spike are observed in the graph to indicate nomination period and ceremony period. There are a period of silence between the two periods. The frequency pattern keeps unchanged each year. We show three years (2007,8,9) in the graph. By making use of recurrent event queries’ periodic properties, we calculated the query period as a new feature.

We use autocorrelation to calculate the period.

$$R(\tau) = \frac{\sum_{t=1}^{N-\tau}(x_t - \mu)(x_{t+\tau} - \mu)}{\{\sum_{t=1}^{N-\tau}(x_t - \mu)^2(x_{t+\tau} - \mu)^2\}^{1/2}}$$

where $x(t)$ is query daily frequency. N is the number of days used for this query. We can get maximum of 3 years data for some queries but only a few months for others. $R(\tau)$ is autocorrelation function. Peaks (the local biggest $R(\tau)$ given a time window) can be detected from $R(\tau)$ plot. The period T is calculated as the duration between two neighbor peaks. $T = 365$ for the query, “Oscar”. The bottom graph of Fig. 2 shows the autocorrelation function plot for the query Oscar.

3.6 Recurrent event seed word list

Many recurrent event queries share some common words that have recurrent time sense. We list most frequently used recurrent seeds in Table 1. Those seeds are likely combined with other words to form new recurrent event queries. For example, the seed, “new”, can be used by queries “new bmw cars”, “whitney houston new songs”, “apple new iphone”, or “hairstyle new”.

new	results	top	schedule
football	festival	movie	world
show	day	best	tax
result	calendar	honda	ford
download	exam	nfl	miss
awards	toyota	tour	sale
american	fair	list	pictures
election	game	basketball	cup

Table 1: Top recurrent event seed words

To generate the seed list, we tokenized all the queries from Implicit Timestamp queries and split all the tokens. We then sort and unique all the tokens, and submit top tokens to professional editors who are asked to pick 8,000 seeds from the top frequent tokens. Some top tokens were removed if they are not qualified to form recurrent event queries. The editors took about four days to do the judgment according to the token’s time sense and examples of recurrent event queries. However, this is a one-time effort. A token will be in the seed if there are many recurrent event examples formed by this token, by editors’ judgment.

Table 1 shows 32 top seeds. Some seeds connect with time such as, “new, schedule, day, best, calendar”; some relate to sports, “football, game, nfl, tour, basketball, cup”; some about cars, “honda, ford, toyota”. The reason why “miss” is in the seeds is that there are many annual events about beauty contest such as “miss america, miss california, miss korea”.

We use the seed list to generate the following three features:

AveNumberTokensSeeds: number of tokens that is in the seed list divided by number of tokens in the query.

AveNumberTokensNotSeeds: number of tokens that is not in the seed list divided by number of tokens in the query.

DiffNumberTokensSeeds: The difference of the above two values.

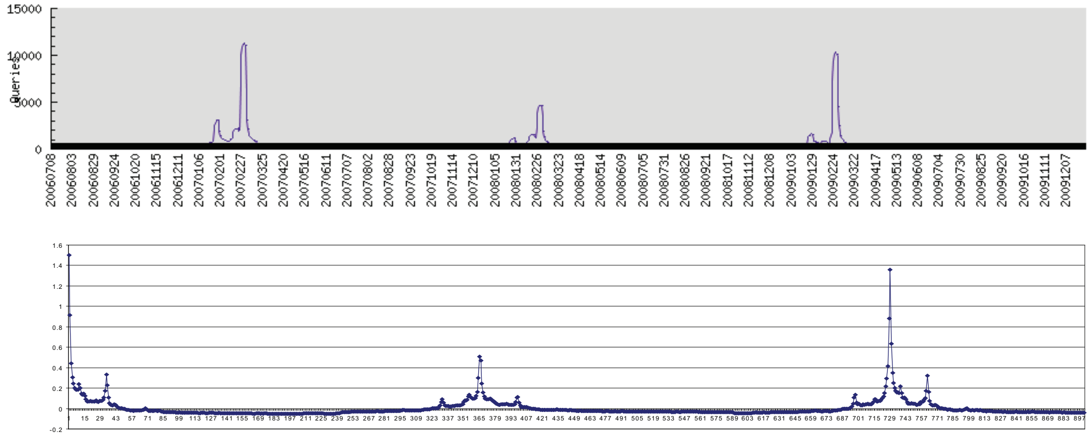


Figure 2: Frequency waveform(top) and corresponding autocorrelation curve (bottom) for query *Oscar*.

4 Learning Approach for REQ

The REQ classification is a typical machine learning task. Given M observed samples used for training data, $\{(\mathbf{x}_0, y_0), (\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_M, y_M)\}$ where $\mathbf{x}_i$ is a feature vector we developed in last section for a given query. y_i is the observation value, $\{+1, -1\}$, indicating the class of REQ and non-REQ. The task is to find the class probability given an unknown feature vector, $\mathbf{x}'$, that is,

$$p(y = c|\mathbf{x}'), \quad c = +1, -1. \tag{1}$$

There are a lot of machine learning methods applicable to implement Eq. 1. In this work, we adopted three representative methods.

The first method is Naive Bayes method. This method treats features independent. If $\mathbf{x}$ is enxtended into feature vector, $\mathbf{x} = \{x_0, x_1, \cdots, x_N\}$ then,

$$p(y = c|\mathbf{x}) = \frac{1}{Z} p(c) \prod_{i=0}^{i=N} p(x_i|c)$$

The second method is SVM. In this work we used the tool for our experiments, LIBSVM (Chang and Lin, 2001). Because SVM is a well known approach and widely used in many classification task, we skip to describe how to use this tool. Readers can turn to the reference for more details.

The third method is based on decision tree based logistic regression model. The probability is given by the formula below,

$$p(y = c|\mathbf{x}) = \frac{1}{1 + e^{-f(\mathbf{x})}} \tag{2}$$

We employ Gradient Boosted Decision Tree algorithm (Friedman, 2001) to learn the function $f(X)$. Gradient Boosted Decision Tree is an additive regression algorithm consisting of an ensemble of trees, fitted to current residuals, gradients of the loss function, in a forward step-wise manner. It iteratively fits an additive model as

$$f_t(x) = T_t(x; \Theta) + \lambda \sum_{t=1}^{T} \beta_t T_t(x; \Theta_t)$$

such that certain loss function $L(y_i, f_T(x + i))$ is minimized, where $T_t(x; \Theta_t)$ is a tree at iteration t, weighted by parameter β_t, with a finite number of parameters, Θ_t and λ is the learning rate. At iteration t, tree $T_t(x; \beta)$ is induced to fit the negative gradient by least squares.
The optimal weights of trees β_t are determined by

$$\beta_t = \operatorname{argmin}_\beta \sum_{i}^{N} L(y_i, f_{t-1}(x_i) + \beta T(x_i, \theta))$$

Each node in the trees represents a split on a feature. The tuneable parameters in such a machine-learnt model include the number of leaf nodes in each tree, the relative contribution of score from each tree called the shrinkage, and total number of shallow decision trees.

The relative importance of a feature S_i, in such forests of decision trees, is aggregated over all the *m* shallow decision trees (Breiman et al., 1984) as follows:

$$S_i^2 = \frac{1}{M}\sum_{m=1}^{M}\sum_{n=1}^{L-1}\frac{w_l * w_r}{w_l + w_r}(y_l y_r)^2 I(v_t = i) \quad (3)$$

where v_t is the feature on which a split occurs, y_l and y_r are the mean regression responses from the right, and left sub-tree, and w_l and w_r are the corresponding weights to the means, as measured by the number of training examples traversing the left and right sub-trees.

5 REQ Learner Evaluation

We collected 6,000 queries labeled as either Recurrent or Non-recurrent by professional human editors. The 6,000 queries were sampled from Implicit Timestamp queries according to frequency distribution to be representative. We split the queries into 5,000 for training and 1,000 for test. For each query, we calculated features' values as described in Section 3.

The Naive Bayes method used single Gaussian function for each independent feature. Mean and variance were calculated from the training data.

As for LIBSVM, we used C-SVC, linear function as kernel and 1.0 of shrinkage.

The parameters used in the regression model were 20 of trees, 20 of nodes and 0.8 of learning rate (shrinkage).

The test results are shown in Fig. 3, recall-precision curve. We set a series of threshold to the probability of $c = +1$ calculated by Eq. 1 so that we can get the point values of recall and precision in Fig. 3. For example, if we set a threshold of 0.6, a query with a probability larger than 0.6 is classified as REQ. Otherwise, it is non-REQ. The precision is a measure of correctly classified REQ queries divided by all classified REQ queries. The recall is a measure of correctly classified REQ queries divided by all REQ queries in test data.

In addition to the three plots, we also show the results using only one feature, ExplicitQueryRatio, for comparison with the classification method used by (Zhang et al., 2009).All the three models using all features performed better than the existing method using ExplicitQueryRatio. The highest improvement was achieved by GBDT regression tree model. The results of Naive Bayes were lower than SVM and GBDTree. This model is weaker because it treats features independently. Typically SVMs and GBDT gives comparable results on a large class of problems. Since for this task we use features from different sources, the feature values are designed to have larger dynamic range, which is better handled by GBDT.

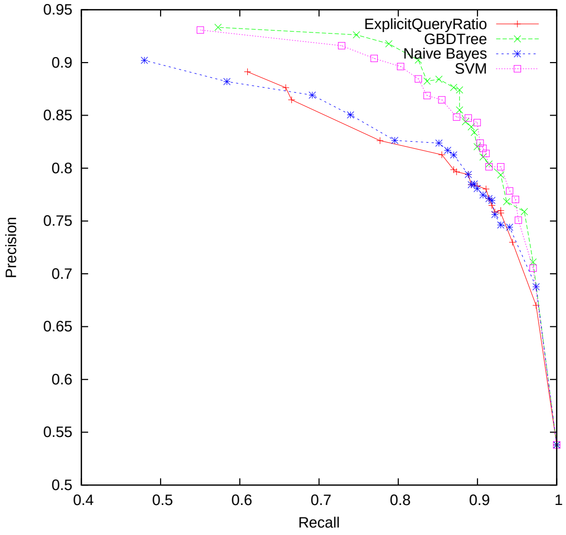


Figure 3: Comparison of precision and recall rate between our method and the existing method.

The features' importance ranked by Equation 3 is shown in Table 2. We list the top 10 features. The No.1 important feature is ExplicitQueryRatio. The second and seventh features are from search session analysis by counting users who changed queries from Implicit Timestamp to Explicit Timestamp. This is a strong source of features. The time series analysis feature is ranked No.3. Calculation of this feature needs two years query log to be much more effective, but we didn't get so large data for many queries. One of the features from recurrent event seed list is ranked No.4. This is also an important feature source. The ChiSquareYearDist feature is ranked 5th, that proves the recurrent event query frequency has a statistical distribution pattern over years. TitleYearTop30 and TitleYearTop10 that are derived from scraping results are ranked the 9th and 10th important.

Fig. 4 shows the distribution of feature values for

Feature	Rank	Score
ExplicitQueryRatio	1	100
NormalizedUserSwitch	2	71.7
AutoCorrelation	3	54.0
AveNumberTokenSeeds	4	48.7
ChiSquareYearDist	5	36.3
YearUrlFPCTR	6	19.1
UserSwitch	7	11.7
QueryDailyFreq	8	10.7
TitleYearTop30	9	10.6
TitleYearTop10	10	5.8

Table 2: Top 10 most important features: rank and importance score (100 is maximum)

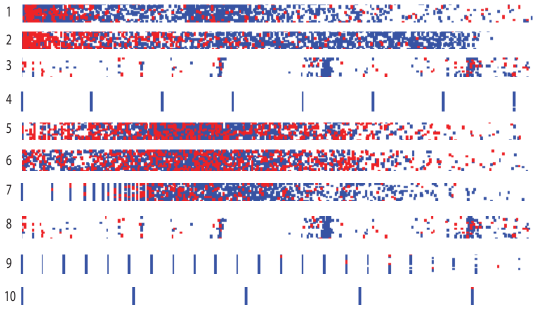


Figure 4: Feature value distribution of all data (blue=REQ, red=non-REQ)

each sample of the 6,000 data, where each point represents a query and each line represents a feature's value for all queries. One point is a query. The features are ordered according to feature importance of Table 2. The "blue" points indicate REQ queries and the "red" points, non-REQ queries. Some features are continuous like the 1st and 2nd. Some feature values are discrete like the last two indicating TitleYearTop30 and TitleYearTop10. There are "red" samples in the 4th feature but overlapped with and covered by "blue" samples visually.

In the Table 3, we show F-Measure values as we gradually added features from the feature, ExplicitQueryRatio, according to feature importance in Table 2. We listed the F-Measure values under three threshold, 0.6, 0.7 and 0.8. Higher threshold will increase classifier precision rate but reduce recall rate. F-Measure is a metric combining precision rate and recall rate. It is clearly observed that the classifier performance is improved as more features are used.

	Threshold		
Feature	0.6	0.7	0.8
ExplicitQueryRatio	0.833	0.833	0.752
+NormalizedUserSwitch	0.840	0.837	0.791
+AutoCorrelation	0.850	0.839	0.823
+AveNumberTokenSeeds	0.857	0.854	0.834
+ChiSquareYearDist	0.857	0.864	0.839
+YearUrlFPCTR	0.869	0.867	0.837
+UserSwitch	0.862	0.862	0.846
+QueryDailyFreq	0.860	0.852	0.847
+TitleYearTop30	0.854	0.853	0.843
+TitleYearTop10	0.858	0.861	0.852
+All	0.876	0.867	0.862

Table 3: F-Measures as varying thresholds by adding top features.

Query	Probability
ncaa men's basketball tournament	0.999
bmw 328i sedan reviews	0.999
new apple iphone release	0.932
sigir	0.920
new york weather in april	0.717
academy awards reviews	0.404
google ipo	0.120
adidas jp	0.082

Table 4: Probabilities of example queries by GBDT tree classifier

Some query examples, and their scores from our model are listed in Table 4. The last two examples, *google ipo* and *adidas jp*, have very low values, and are not REQs. The first four queries are typical REQs. They have higher values of features ExplicitQueryRatio,Normalized UserSwitch and YearUrlFPCTR. Although both *new apple iphone release reviews* and *academy awards reviews* are about reviews, *academy awards reviews* has lower value of NormalizedUserSwitch and ChiSquareYearDist could be the reason for a lower score.

6 Web Search Ranking

In this section, we use the approach proposed by (Zhang et al., 2009) to test the REQ classifier for Web search ranking. In their approach, search ranking is altered by boosting pages with most recent year if the query is a REQ. The year indicator

		DCG@5			DCG@1		
bucket	#(query)	Organic	Our's	% over Organic	Organic	Ours	% over Organic
[0.0,0.1]	59	6.87	6.96	1.48(-2.3)	4.08	4.19	2.69(-1.07)
[0.1,0.2]	76	5.86	6.01	2.52(0.98)	2.88	2.91	1.14(1.69)
[0.2,0.3]	85	6.33	6.41	1.24(2.12)	3.7	3.7	0.0(0.8)
[0.3,0.4]	75	5.18	5.24	1.18(-0.7)	2.92	2.95	1.14(1.37)
[0.4,0.5]	78	4.96	4.82	-2.84(-1.35)	2.5	2.42	-3.06(0)
[0.5,0.6]	84	5.4	5.37	-0.45(-0.3)	2.82	2.85	1.05(-1.5)
[0.6,0.7]	78	4.78)	5.19)	8.42(3.64)	2.56	2.83	10.75(4.1)
[0.7,0.8]	80	4.45	4.60	3.41(3.19)	2.21	2.26	1.98(2.8)
[0.8,0.9]	78	4.81	4.96	3.15(4.79)	2.32	2.33	0.55(0.65)
[0.9,1.0]	107	5.08	5.50	**8.41***(4.41)	2.64	3.09	**16.78***(1.36)
[0.0,1.0]	800	5.33	5.47	**2.74***(2.17)	2.83	2.93	**3.6***(1.26)

Table 5: REQ learner improves search engine organic results. The numbers in the brackets are by Zhang's methods. Direct comparison with Zhang's method is valid only in the last line, using all queries. A sign “$*$” indicates statistical significance (p-value<0.05)

can be detected either from title or URL of the result. For clarity, we re-write their ranking function as below,

$$F(q,d) = R(q,d) + [e(d_o, d_n) + k]e^{\lambda\alpha(q)}$$

where the ranking function, $F(q,d)$, consists of two parts: the base function $R(q,d)$ plus boosting. If the query q is not a REQ, boosting is set to zero. Otherwise, boosting is decided by $e(d_o, d_n)$, k, λ and $\alpha(q)$. $e(d_o, d_n)$ is the difference of base ranking score between the oldest page and the newest page. If the newest page has a lower ranking score than the oldest page, then the difference is added to the newest page to promote the ranking of the newest page.

$\alpha(q)$ is the confidence score of a REQ query. It is the value of Eq. 1. λ and k are two empirical parameters. (Zhang et al., 2009)'s work has experimented the effects of using different value of λ and k ($\lambda = 0$ equals to no discounts for ranking adjustment). We used $\lambda = 0.4$ and $k = 0.3$ which were the best configuration in (Zhang et al., 2009).

For evaluating our methods, we randomly extracted 800 queries from the Implicit Timestamp queries. We scraped a commercial search engine using the 800 queries. We extracted the top five search results for each query under three configures: organic search engine results, (Zhang et al., 2009)'s method and ours using REQ classifier. We asked human editors to judge all the scraped (query, url) pairs. Editors assign five grades according to relevance between query and articles: Perfect, Excellent, Good, Fair, and Bad. For example, a “Perfect” grade means the content of the url match exactly the query intent.

We use Discounted Cumulative Gain (DCG) (Jarvelin and Kekalainen, 2002) at rank k as our primary evaluation metrics to measure retrieval performance. DCG is defined as,

$$DCG@k = \sum_{i=1}^{k} \frac{2^{r(i)} - 1}{log_2(1+i)}$$

where $r(i) \in \{0 \dots 4\}$ is the relevance grade of the ith ranked document.

The Web search ranking results are shown in Table 5. We used GBDT tree learning methods because it achieved the best results. We divided 800 test queries into 10 buckets according to the classifier probability. The bucket, [0.0,0.1], contains the query with a classifier probability greater than 0 but less than 0.1. Our results are compared with organic search results, but we also show the improvements over search organic by (Zhang et al., 2009) in the brackets. Because Zhang's approach output different classifier values from Ours for the same query, buckets of the same range in the Table contain different queries. Hence, it is inappropriate to compare

Zhang's with Ours for the same buckets except the last row where we used all the queries.

Our classifier's overall performance is much better than the organic search results. We achieved 2.74% DCG@5 gain and 3.6% DCG@1 gain over organic search for all queries. The gains are higher than (Zhang et al., 2009)'s results with regards to improvement over organic results. By direct comparison, Ours was 2.7% better than Zhangs significantly in terms of DCG@1 by Wilcoxon significant test. DCG@5 is 1.1% better, but not significant. The table also show that the higher buckets with higher probability achieved higher DCG gain than the lower buckets overall. Our approach observed 16.78% DCG@1 gain for bucket [0.9,1.0]. This shows that our methods are very effective.

7 Conclusions

We found most of REQ are long tail queries that pose a major challenge to Web search. We have demonstrated learning REQ is important for Web search. this type of queries can't be solved in traditional ranking method. We found building a REQ classifier was a good solution. Our work described using machine learning method to build REQ classifier. Our proposed methods are novel comparing with traditional query classification methods. We identified and developed features from query log, search session, click and time series analysis. We applied several ML approaches including Naive Bayes, SVM and GBDT tree to implement REQ learner. Finally, we show through ranking experiments that the methods we proposed are very effective and beneficial for search engine ranking.

Acknowledgements

We express our thanks to who have assisted us to complete this work, especially, to Fumiaki Yamaoka, Toru Shimizu, Yoshinori Kobayashi, Mitsuharu Makita, Garrett Kaminaga, Zhuoran Chen.

References

R. Baeza-Yates, F. Saint-Jean, and C. Castillo. 2002. Web dynamics, age and page qualit. *String Processing and Information Retrieval*, pages 453–461.

Steven M. Beitzel, Eric C. Jensen, Ophir Frieder, David Grossman, David D. Lewis, Abdur Chowdhury, and Aleksandr Kolcz. 2005. Automatic web query classification using labeled and unlabeled training data. In *SIGIR '05*, pages 581–582.

K. Berberich, M. Vazirgiannis, and G. Weikum. 2005. Time-aware authority rankings. *Internet Math*, 2(3):301–332.

L. Breiman, J. Friedman, R. Olshen, and C. Stone. 1984. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA.

S. Brin and L. Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Proceedings of International Conference on World Wide Web*.

Andrei Z. Broder, Marcus Fontoura, Evgeniy Gabrilovich, Amruta Joshi, Vanja Josifovski, and Tong Zhang. Robust classification of rare queries using web knowledge. In *SIGIR '07*, pages 231–238.

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines*. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

J. Cho, S. Roy, and R. Adams. 2005. Page quality: In search of an unbiased web ranking. *Proc. of ACM SIGMOD Conference*.

F. Diaz. 2009. Integration of news content into web results. *Proceedings of the Second ACM International Conference on Web Search and Data Mining (WSDM)*, pages 182–191.

Anlei Dong, Yi Chang, Zhaohui Zheng, Gilad Mishne, Jing Bai, Ruiqiang Zhang, Karolina Buchner, Ciya Liao, and Fernando Diaz. 2010a. Towards recency ranking in web search. *Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM)*, pages 11–20.

Anlei Dong, Ruiqiang Zhang, Pranam Kolari, Jing Bai, Fernando Diaz, Yi Chang, Zhaohui Zheng, and Hongyuan Zha. 2010b. Time is of the essence: improving recency ranking using twitter data. *19th International World Wide Web Conference (WWW)*, pages 331–340.

Jonathan L. Elsas and Susan T. Dumais. 2010. Leveraging temporal dynamics of document content in relevance ranking. In *WSDM*, pages 1–10.

J. H. Friedman. 2001. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232.

Kalervo Jarvelin and Jaana Kekalainen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20:2002.

K. Sparck Jones, S. Walker, and S. E. Robertson. 2000. A probabilistic model of information retrieval: development and comparative experiments. *Inf. Process. Manage.*, 36(6):779–808.

A. C. Knig, M. Gamon, and Q. Wu. 2009. Click-through prediction for news queries. *Proc. of SIGIR*, pages 347–354.

Ying Li, Zijian Zheng, and Honghua (Kathy) Dai. 2005. Kdd cup-2005 report: facing a great challenge. *SIGKDD Explor. Newsl.*, 7(2):91–99.

Xiao Li, Ye yi Wang, and Alex Acero. 2008. Learning query intent from regularized click graphs. In *In SIGIR 2008*, pages 339–346. ACM.

Donald Metzler, Rosie Jones, Fuchun Peng, and Ruiqiang Zhang. 2009. Improving search relevance for implicitly temporal queries. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 700–701.

S. Nunes. 2007. Exploring temporal evidence in web information retrieval. *BCS IRSG Symposium: Future Directions in Information Access*.

S. Pandey, S. Roy, C. Olston, J. Cho, and S. Chakrabarti. 2005. Shuffling a stacked deck: The case for partially randomized ranking of search engine results. *VLDB*.

G. Salton and M. J. McGill. 1983. *Introduction to modern information retrieval*. McGraw-Hill, NY.

Dou Shen, Rong Pan, Jian-Tao Sun, Jeffrey Junfeng Pan, Kangheng Wu, Jie Yin, and Qiang Yang. 2005. Q2c@ust: our winning solution to query classification in kddcup 2005. *SIGKDD Explor. Newsl.*, 7(2):100–110.

Dou Shen, Rong Pan, Jian-Tao Sun, Jeffrey Junfeng Pan, Kangheng Wu, Jie Yin, and Qiang Yang. 2006. Query enrichment for web-query classification. *ACM Trans. Inf. Syst.*, 24(3):320–352.

Ruiqiang Zhang, Yi Chang, Zhaohui Zheng, Donald Metzler, and Jian-yun Nie. 2009. Search result re-ranking by feedback control adjustment for time-sensitive query. In *HLT-NAACL '09*, pages 165–168.

Staying Informed: Supervised and Semi-Supervised Multi-view Topical Analysis of Ideological Perspective

Amr Ahmed
School of Computer Science
Carnegie Mellon University
amahmed@cs.cmu.edu

Eric P. Xing
School of Computer Science
Carnegie Mellon University
epxing@cs.cmu.edu

Abstract

With the proliferation of user-generated articles over the web, it becomes imperative to develop automated methods that are aware of the ideological-bias implicit in a document collection. While there exist methods that can classify the ideological bias of a given document, little has been done toward understanding the nature of this bias on a topical-level. In this paper we address the problem of modeling ideological perspective on a topical level using a factored topic model. We develop efficient inference algorithms using Collapsed Gibbs sampling for posterior inference, and give various evaluations and illustrations of the utility of our model on various document collections with promising results. Finally we give a Metropolis-Hasting inference algorithm for a semi-supervised extension with decent results.

1 Introduction

With the avalanche of user-generated articles over the web, it is quite important to develop models that can recognize the ideological bias behind a given document, summarize where this bias is manifested on a topical level, and provide the user with alternate views that would help him/her staying informed about different perspectives. In this paper, we follow the notion of ideology as defines by Van Dijk in (Dijk, 1998) as “a set of general abstract beliefs commonly shared by a group of people.” In other words, an ideology is a set of ideas that directs one's goals, expectations, and actions. For instance, *freedom of choice* is a general aim that directs the actions of“liberals”, whereas *conservation of values* is the parallel for “conservatives”.

We can attribute the lexical variations of the word content of a document to three factors:

- **Writer Ideological Belief.** A liberal writer might use words like freedom and choice regardless of the topical content of the document. These words define the abstract notion of belief held by the writer and its frequency in the document largely depends on the writer's style.
- **Topical Content.** This constitutes the main source of the lexical variations in a given document. For instance, a document about abortion is more likely to have facts related to abortion, health, marriage and relationships.
- **Topic-Ideology Interaction.** When a liberal thinker writes about abortion, his/her abstract beliefs are materialized into a set of concrete opinions and stances, therefore, we might find words like: pro-choice and feminism. On the contrary, a conservative writer might stress issues like pro-life, God and faith.

Given a collection of ideologically-labeled documents, our goal is to develop a computer model that *factors* the document collection into a representation that reflects the aforementioned three sources of lexical variations. This representation can then be used for:

- **Visualization.** By visualizing the abstract notion of belief in each ideology, and the way each ideology approaches and views mainstream topics, the user can view and contrast each ideology side-by-side and build the right mental landscape that acts as the basis for his/her future decision making.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1140–1150,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

- **Classification or Ideology Identification.** Given a document, we would like to tell the user from which side it was written, and explain the ideological bias in the document at a topical level.

- **Staying Informed: Getting alternative views[1].** Given a document written from perspective A, we would like the model to provide the user with other documents that represent alternative views about the same topic addressed in the original document.

In this paper, we approach this problem using Topic Models (Blei et al., 2003). We introduce a factored topic model that we call multi-view Latent Dirichlet Allocation or mview-LDA for short. Our model views the word content of each document as the result of the interaction between the document's idealogical and topical dimensions. The rest of this paper is organized as follows. First, in Section 2, we review related work, and then present our model in Section 3. Then in Section 4, we detail a collapsed Gibbs sampling algorithm for posterior inference. Sections 5 and 6 give details about the dataset used in the evaluation and illustrate the capabilities of our model using both qualitative and quantitative measures. Section 7 describes and evaluates the efficacy of a semi-supervised extension, and finally in Section 8 we conclude and list several directions for future research.

2 Related Work

Ideological text is inherently subjective, thus our work is related to the growing area of subjectivity analysis(Wiebe et al., 2004; Riloff et al., 2003). The goal of this area of research is to learn to discriminate between subjective and objective text. In contrast,in modeling ideology, we aim toward contrasting two or more ideological perspectives each of which is subjective in nature. Further more, subjective text can be classified into sentiments which gave rise to a surge of work in automatic opinion mining (Wiebe et al., 2004; Yu and Hatzivassiloglou, 2003; Pang et al., 2002; Turney and Littman, 2003; Popescu and Etzioni, 2005) as well as sentiment analysis and product review mining (Nasukawa and Yi, 2003; Hu and Liu, 2004; Pang and Lee, 2008; Branavan et al., 2008; Titov and McDonald, 2008; Titov and McDonald, 2008; Mei et al., 2007; Ling et al., 2008). The research goal of sentiment analysis and classification is to identify language used to convey positive and negative opinions, which differs from contrasting two ideological perspectives. While ideology can be expressed in the form of a sentiment toward a given topic,like abortion, ideological perspectives are reflected in many ways other than sentiments as we will illustrate later in the paper. Perhaps more related to this paper is the work of (Fortuna et al., 2008; Lin et al., 2008) whose goal is to detect bias in news articles via discriminative and generative approaches, respectively. However, this work still addresses ideology at an abstract level as opposed to our approach of modeling ideology at a topical level. Finally, independently, (Paul and Girju, 2009) gives a construction similar to ours however for a different task [2].

3 Multi-View Topic Models

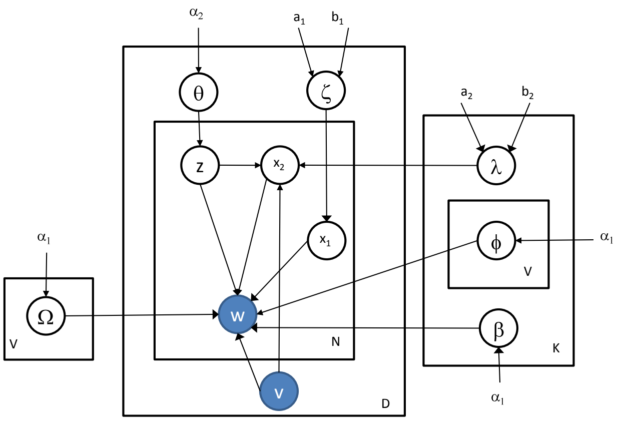
In this section we introduce multi-view topic models, or mview-LDA for short. Our model, mview-LDA, views each document as the result of the interaction between its topical and idealogical dimensions. The model seeks to explain lexical variabilities in the document by attributing this variabilities to one of those dimensions or to their interactions. Topic models, like LDA, define a generative process for a document collection based on a set of parameters. LDA employs a semantic entity known as *topic* to drive the generation of the document in question. Each topic is represented by a topic-specific word distribution which is modeled as a multinomial distribution over words, denoted by $\mathrm{Multi}(\beta)$. The generative process of LDA proceeds as follows:

1. Draw topic proportions $\theta_d|\alpha \sim \mathrm{Dir}(\alpha)$.
2. For each word
   (a) Draw a topic $z_n|\theta_d \sim \mathrm{Mult}(\theta_d)$.
   (b) Draw a word $w_n|z_n, \beta \sim \mathrm{Multi}(\beta_{z_n})$.

In step 1 each document d samples a topic-mixing vector θ_d from a Dirichlet prior. The component $\theta_{d,k}$

[1] In this paper, we use the words ideology, view, perspective interchangeably to denote the same concept

[2] In fact, we only get to know about this related work after our paper was accepted



Variable	Meaning
w	word
v	document's ideology
z	topic
x_1, x_2	word switches, one per word (see text)
θ	document-specific distribution over topics
ξ	document's expected usage of the ideology's background topic
Ω	ideology's background-topic
β	ideology-independent topic distribution
ϕ	ideology-specific topic distribution
λ	topic bias across ideology

Figure 1: A plate diagram of the graphical model.

of this vector defines how likely topic k will appear in document d. For each word in the document w_n, a topic indicator z_n is sampled from θ_d, and then the word itself is sampled from a topic-specific word distribution specified by this indicator. Thus LDA can capture and represent lexical variabilities via the components of θ_d which represents the topical content of the document. In the next section we will explain how our new model mview-LDA can capture other sources of lexical variabilities beyond topical content.

3.1 Multi-View LDA

As we noted earlier, LDA captures lexical variabilities due to topical content via θ_d and the set of topics $\beta_{1:K}$. In mview-LDA each document d is tagged with the ideological view it represents via the observed variable v_d which takes values in the discrete range: $\{1, 2, \cdots, V\}$ as shown in Fig. 1. For simplicity, lets first assume that $V = 2$. The topics $\beta_{1:K}$ retain the same meaning: a set of K multinomial distributions each of which represents a given theme or factual topic. In addition, we utilize an ideology-specific topic Ω_v which is again a multinomial distribution over the same vocabulary. Ω_v models the abstract belief shared by all the documents written from view v. In other words, if v denotes the liberal perspective, then Ω_v gives high probability to words like progressive, choice, etc. Moreover, we defined a set of $K \times V$ topics that we refer to as ideology-specific topics. For example, topic $\phi_{v,k}$ represents how ideology v addresses topic k. The generative process of a document d with ideological view v_d proceeds as follows:

```
1. Draw $\xi_d \sim \mathrm{Beta}(a_1, b_1)$
2. Draw topic proportions $\theta_d|\alpha \sim \mathrm{Dir}(\alpha_2)$.
3. For each word $w_n$
   (a) Draw $x_{n,1} \sim \mathrm{Bernoulli}(\xi_d)$
   (b) If$(x_{n,1} = 1)$
       i. Draw $w_n|x_{n,1} = 1 \ \sim\ \mathrm{Multi}(\Omega_{v_d})$
   (c) If$(x_{n,1} = 0)$
       i. Draw $z_n|\theta_d \sim \mathrm{Mult}(\theta_d)$.
       ii. Draw $x_{n,2}|v_d, z_n \sim \mathrm{Bernoulli}(\lambda_{z_n})$
       iii. If$(x_{n,2} = 1)$
           A. Draw $w_n|z_n, \beta \sim \mathrm{Multi}(\beta_{z_n})$.
       iv. If$(x_{n,2} = 0)$
           A. Draw $w_n|v_d, z_n \sim \mathrm{Multi}(\phi_{v_d,z_n})$.
```

In step 1, we draw a document-specific biased coin,ξ_d. The bias of this coin determines the proportions of words in the document that are generated from its ideology background topic Ω_{v_d}. As in LDA, we draw the document-specific topic proportion θ_d from a Dirichlet prior. θ_d thus controls the lexical variabilities due to topical content inside the document.

To generate a word w_n, we first generate a coin flip $x_{n,1}$ from the coin ξ_d. If it turns head, then we proceed to generate this word from the ideology-specific topic associated with the document's ideological view v_d. In this case, the word is drawn independently of the topical content of the document, and thus accounts for the lexical variation due to the ideology associated with the document. The proportion of such words is document-specific by design and depends on the writer's style to a large degree. If $x_{n,1}$ turns to be tail,we proceed to the next step and draw a topic-indicator z_n. Now, we have two choices: either to generate this word directly from the ideology-independent portion of the topic β_{z_n}, or to draw the word from the ideology-specific portion ϕ_{v_d,z_n}. The choice here is **not** document specific, but rather depends on the interaction between the ideology and the specific topic in question. If the ideology associated with the document holds a strong opinion or view with regard to this topic, then we expect that most of the time we will take the second choice, and generate w_n from ϕ_{v_d,z_n}; and vice versa. This decision is controlled by the Bernoulli variable λ_{z_n}. Therefore, in step $c.ii$, we first generate a coin flip $x_{n,2}$ from λ_{z_n}. Based on $x_{n,2}$ we either generate the word from the ideology-independent portion of the topic β_{z_n}, and this constitutes how the model accounts for lexical variation due to the topical content of the document, or generate the word from the ideology-specific portion of the topic ϕ_{v_d,z_n}, and this specifies how the model accounts for lexical variation due to the interaction between the topical and ideological dimensions of the document.

Finally, it is worth mentioning that the decision to model λ_{z_n}[3] at the topic-ideology level rather than at the document level, as we have done with ξ_d, stems from our goal to capture ideology-specific behavior on a corpus level rather than capturing document-specific writing style. However, it is worth mentioning that if one truly seeks to measure the degree of bias associated with a given document,then one can compute the frequency of the event $x_{n,2} = 0$ from posterior samples. In this case, λ_{z_n} acts as the prior bias only. Moreover, computing the frequency of the event $x_{n,2} = 0$ and $z_n = k$ gives the document's bias toward topic k per se.

Finally, it is worth mentioning that all multinomial topics in the model: β, Ω, ϕ are generated once for the whole collection from a symmetric Dirichlet prior, similarly, all bias variables, $\lambda_{1:K}$ are sampled from a Beta distribution also once at the beginning of the generative process.

4 Posterior Inference Via Collapsed Gibbs Sampling

The main tasks can be summarized as follows:

- **Learning**: Given a collection of documents, find a point estimate of the model parameters (i.e. $\beta, \Omega, \phi, \lambda$,etc.).
- **Inference**: Given a new document, and a point estimate of the model parameters, find the posterior distribution of the latent variables associated with the document at hand: $(\theta_d, \{x_{n,1}\}, \{z_n\}, \{x_{n,2}\})$.

Under a hierarchical Bayesian setting, like the approach we took in this paper, both of these tasks can be handled via posterior inference. Under the generative process, and hyperparmaters choices, outlined in section 3, we seek to compute:

$$P(\mathbf{d}_{1:D}, \beta_{1:K}, \mathbf{\Omega}_{1:V}, \phi_{\mathbf{1:V},\mathbf{1:K}}, \lambda_{1:K} | \alpha, a, b, \mathbf{w}, \mathbf{v}),$$

where $\mathbf{d}$ is a shorthand for the hidden variables $(\theta_d, \xi_d, \mathbf{z}, \mathbf{x_1}, \mathbf{x_2})$ in document d. The above posterior probability is unfortunately intractable,and we approximate it via a collapsed Gibbs sampling procedure (Griffiths and Steyvers, 2004; Gelman et al., 2003) by integrating out, i.e. collapsing, the following hidden variables: the topic-mixing vectors θ_d and the ideology bias ξ_d for each document, as well as all the multinomial topic distributions: $(\beta, \Omega$ and $\phi)$ in addition to the ideology-topic biases given by the set of λ random variables.

Therefore, the state of the sampler at each iteration contains only the following topic indicators and coin flips for each document:$(\mathbf{z}, \mathbf{x_1}, \mathbf{x_2})$. We alternate sampling each of these variables conditioned on its Markov blanket until convergence. At convergence, we can calculate expected values for all the parameters that were integrated out, especially for the topic distributions, for each document's latent representation (mixing-vector) and for all coin biases. To ease the calculation of the Gibbs sampling update equations we keep a set of sufficient statistics (SS) in the form of co-occurrence counts and sum matrices of the form C^{EQ}_{eq} to denote the number of times instance e appeared with instance q. For example, C^{WK}_{wk} gives the number of times word w was sampled from the ideology-independent portion of

[3] In an earlier version of the work we modeled λ on a per-ideology basis, however, we found that using a single shared λ results in more robust results

topic k. Moreover, we follow the standard practice of using the subscript $-i$ to denote the same quantity it is added to without the contribution of item i. For example,$C^{WK}_{wk,-i}$ is the same as C^{WK}_{wk} without the contribution of word w_i. For simplicity, we might drop dependencies on the document whenever the meaning is implicit form the context.

For word w_n in document d, instead of sampling $z_n, x_{n,1}, x_{n,2}$ independently, we sample them as a block as follows:

$$P(x_{n,1} = 1|w_n = w, v_d = v) \propto$$
$$(C^{DX_1}_{d1,-n} + a_1) \times \frac{C^{VW}_{vw,-n} + \alpha_1}{\sum_{w'}(C^{VW}_{vw',-n} + \alpha_1)}$$

$$P(x_{n,1} = 0, x_{2,n} = 1, z_n = k|w_n = w, v_d = v)$$
$$\propto (C^{DX_1}_{d0,-n} + b_1) \times \frac{C^{KX_2}_{k1,-n} + a_2}{C^{KX_2}_{k1,-n} + C^{KX_2}_{k0,-n} + a_2 + b_2}$$
$$\times \frac{C^{KW}_{kw,-n} + \alpha_1}{\sum_{w'}(C^{KW}_{kw',-n} + \alpha_1)} \times \frac{C^{DK}_{dk,-n} + \alpha_2}{\sum_{k'}(C^{DK}_{dk',-n} + \alpha_2)}$$

$$P(x_{n,1} = 0, x_{2,n} = 0, z_n = k|w_n = w, v_d = v)$$
$$\propto (C^{DX_1}_{d0,-n} + b_1) \times \frac{C^{KX_2}_{k0,-n} + b_2}{C^{KX_2}_{k1,-n} + C^{KX_2}_{k0,-n} + a_2 + b_2}$$
$$\times \frac{C^{VKW}_{vkw,-n} + \alpha_1}{\sum_{w'}(C^{VKW}_{vkw',-n} + \alpha_1)} \times \frac{C^{DK}_{dk,-n} + \alpha_2}{\sum_{k'}(C^{DK}_{dk',-n} + \alpha_2)}$$

The above three equations can be normalized to form a $2 * K + 1$ multinomial distribution: one component for generating a word from the ideology topic, K components for generating the word from the ideology-independent portion of topic $k = 1, \cdots, K$, and finally K components for generating the word from the ideology-specific portion of topic $k = 1, \cdots, K$. Each of these $2 * K + 1$ cases corresponds to a unique assignment of the variables $z_n, x_{n,1}, x_{n,2}$. Therefore, our Gibbs sampler just repeatedly draws sample from this $2*K+1$-components multinomial distribution until convergence. Upon convergence, we compute point estimates for all the collapsed variables by a simple marginalization of the appropriate count matrices. During **inference**, we hold the corpus-level count matrices fixed, and keep sampling from the above $2*K+1$-component multinomial while only changing the document-level count matrices: C^{DK}, C^{DX_1} until convergence. Upon convergence, we compute estimates for ξ_d and θ_d by normalizing C^{DK} and C^{DX_1} (or possibly averaging this quantity across posterior samples). As we mentioned in Section 3, to compute the ideology-bias in addressing a given topic say k in a given document, say d, we can simply compute the expected value of the event $x_{n,2} = 0$ and $z_n = k$ across posterior samples.

5 Data Sets

We evaluated our model over three datasets: the bitterlemons croups and a two political blog-data set. Below we give details of each dataset.

5.1 The Bitterlemons dataset

The bitterlemons corpus consists of the articles published on the website `http://bitterlemons.org/`. The website is set up to contribute to mutual understanding between Palestinians and Israelis through the open exchange of ideas. Every week, an issue about the Israeli-Palestinian conflict is selected for discussion, and a Palestinian editor and an Israeli editor contribute one article each addressing the issue. In addition, the Israeli and Palestinian editors invite one Israeli and one Palestinian to express their views on the issue. The data was collected and pre-proceed as describes in (Lin et al., 2008). Overall, the dataset contains 297 documents written from the Israeli's point of view, and 297 documents written from the Palestinian's point of view. On average each document contains around 740 words. After trimming words appearing less than 5 times, we ended up with a vocabulary size of 4100 words. We split the dataset randomly and used 80% of the documents for training and the rest for testing.

5.2 The Political Blog Datasets

The first dataset refereed to as *Blog-1* is a subset of the data collected and processed in (Yano et al., 2009). The authors in (Yano et al., 2009) collected blog posts from blog sites focusing on American politics during the period November 2007 to October 2008. We selected three blog sites from this dataset: the Right Wing News (right-ideology) ; the Carpetbagger, and Daily Kos as representatives

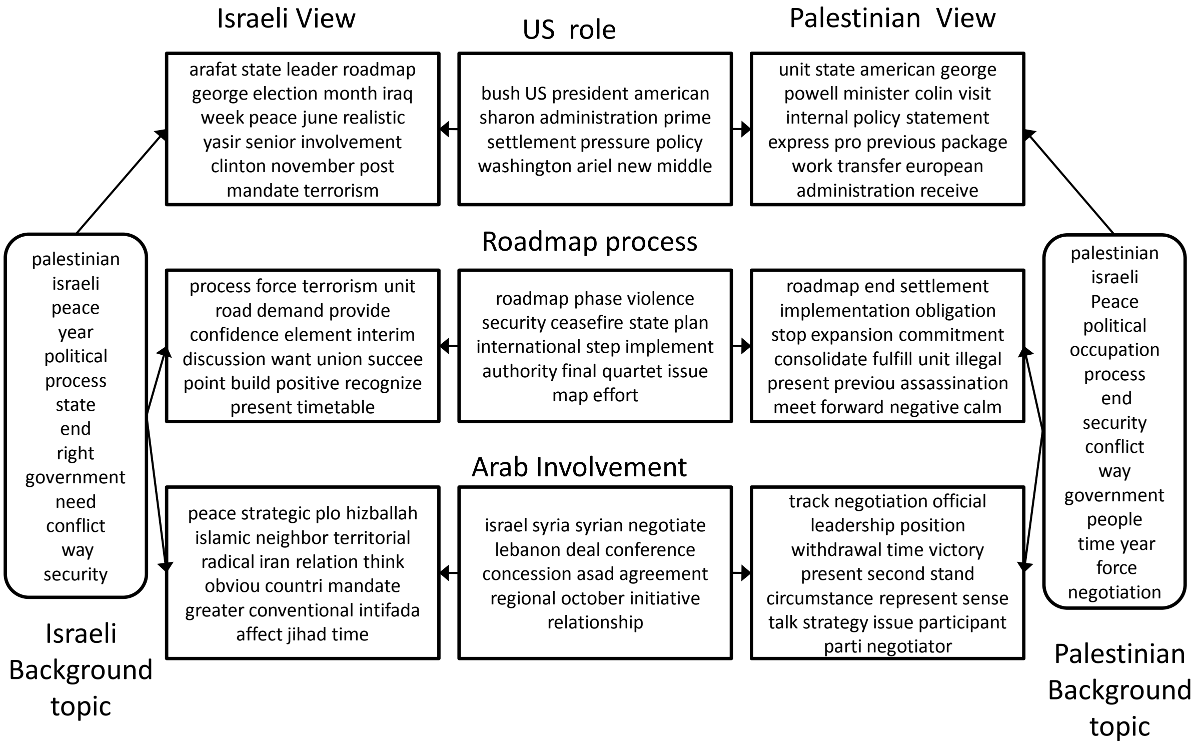

Israeli View
US role
Palestinian View
arafat state leader roadmap george election month iraq week peace june realistic yasir senior involvement clinton november post mandate terrorism
bush US president american sharon administration prime settlement pressure policy washington ariel new middle
unit state american george powell minister colin visit internal policy statement express pro previous package work transfer european administration receive
Roadmap process
process force terrorism unit road demand provide confidence element interim discussion want union succee point build positive recognize present timetable
roadmap phase violence security ceasefire state plan international step implement authority final quartet issue map effort
roadmap end settlement implementation obligation stop expansion commitment consolidate fulfill unit illegal present previou assassination meet forward negative calm
Arab Involvement
peace strategic plo hizballah islamic neighbor territorial radical iran relation think obviou countri mandate greater conventional intifada affect jihad time
israel syria syrian negotiate lebanon deal conference concession asad agreement regional october initiative relationship
track negotiation official leadership position withdrawal time victory present second stand circumstance represent sense talk strategy issue participant parti negotiator
palestinian israeli peace year political process state end right government need conflict way security
Israeli Background topic
palestinian israeli Peace political occupation process end security conflict way government people time year force negotiation
Palestinian Background topic

Figure 2: Illustrating the big picture overview over the bitterlemons dataset using few topics. Each box lists the top words in the corresponding multinomial topic distribution. See text for more details

of the liberal view (left-ideology). After trimming short posts of less than 20 words, we ended up with 2040 posts distributed as 1400 from the left-wing and the rest from the right-wing. On average, each post contains around 100 words and the total size of the vocabulary is 14276 words. For this dataset, we followed the train-test split in (Yano et al., 2009). In this split each blog is represented in both training and test sets. Thus this dataset does not measure the model's ability to generalize to a totally different writing style.

The second dataset refereed to as *Blog-2* is similar to Blog-1 in its topical content and time frame but larger in its blog coverage (Eisenstein and Xing, 2010). Blog-2 spans 6 blogs: three from the left-wing and three from the right-wing. The dataset contains 13246 posts. After removing words that appear less then 20 times, the total vocabulary becomes 13236 with an average of 200 words per post. We used 4 blogs (2 from each view) for training and held two blogs (one from each view) for testing. Thus this dataset measures the model's ability to generalize to a totally new blog.

6 Experimental Results

In this section we gave various qualitative and quantitative evaluations of our model over the datasets listed in Section 5. For all experiments, we set $\alpha_1 = .01, \alpha_2 = .1, a = 1$ and $b = 1$. We run Gibbs sampling during training for 1000 iterations. During inference, we ran Gibbs sampling for 300 iterations, and took 10 samples, with 50-iterations lag, for evaluations.

6.1 Visualization and Browsing

One advantage of our approach is its ability to create a "big-picture" overview of the interaction between ideology and topics. In figure 2 we show a portion of that diagram over the bitterlemons dataset. First note how the ideology-specific topics in both ideology share the top-three words, which highlights that the two ideologies seek peace even though they still both disagree on other issues. The figure gives example of three topics: the US role, the Roadmap peace process, and the Arab involvement in the conflict (the name of these topics were hand-crafted). For each topic, we display the top words in the ideology-independent part of the topic (β), along with top words in each ideology's view of the topic (ϕ).

For example, when discussing the roadmap process, the Palestinian view brings the following issues: [the Israeli side should] implement the obligatory points in this agreement, stop expansion of settlements, and move forward to the commitments brought by this process. On the other hand, the Israeli side brings the following points: [Israelis] need to build confidence [with Palestinian], address the role of terrorism on the implementation of the process, and ask for a positive recognition of Israel from the different Palestinian political parties. As we can see, the ideology-specific portion of the topic **needn't** always represent a **sentiment** shared by its members toward a given topic, but it might rather includes extra important dimensions that need to be taken into consideration when addressing this topic.

Another interesting topic addresses the involvement of the neighboring Arab countries in the conflict. From the Israeli point of view, Israel is worried about the existence of hizballah [in lebanon] and its relationship with radical Iran, and how this might affect the Palestinian-uprising (Intifada) and Jihad. From the other side, the Palestinians think that the Arab neighbors need to be involved in the peace process and negotiations as some of these countries like Syria and Lebanon are involved in the conflict.

The user can use the above chart as an entry point to retrieve various documents pertinent to a given topic or to a given view over a specific topic. For instance, if the user asks for a representative sample of the Israeli(Palestinian) view with regard to the roadmap process, the system can first retrieve documents tagged with the Israeli(Palestinian) view and having a high topical value in their latent representation θ over this topic. Finally, the system then sorts these documents by how much bias they show over this topic. As we discussed in Section 4, this can be done by computing the expected value of the event $x_{n,2} = 0$ and $z_n = k$ where k is the topic under consideration.

6.2 Classification

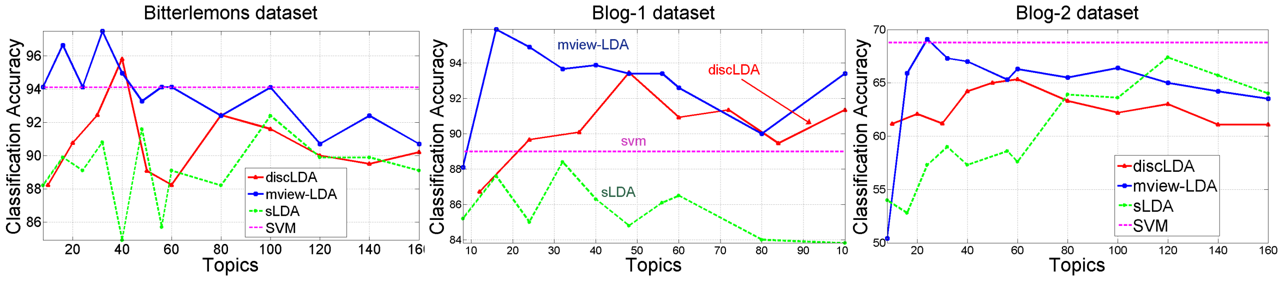
We have also performed a classification task over all the datasets. The Scenario proceeded as follows. We train a model over the training data with various number of topics. Then given a test document, we predict its ideology using the following equation:

$$v_d = \arg max_{v \in V} P(\mathbf{w_d}|v); \qquad (1)$$

We use three baselines. The first baseline is an SVM classifier trained on the normalized word frequency of each document. We trained SVM using a regularization parameter in the range $\{1, 10, 20, \cdots, 100\}$ and report the best result (i.e. no cross-validation was performed). The other two are supervised LDA models: supervised LDA (sLDA) (Wang et. al., 2009; Blei and McCauliffe, 2007) and discLDA (Lacoste-Julien et al., 2008). discLDA is a conditional model that divides the available number of topics into class-specific topics and shared-topics. Since the code is not publicly available, we followed the same strategy in the original paper and share $0.1K$ topics across ideologies and then divide the rest of the topics between ideologies[4]. However, unlike our model, there are no internal relationships between these two sets of topics. The decision rule employed by discLDA is very similar to the one we used for mview-LDA in Eq (1). For sLDA, we used the publicly available code by the authors.

As shown in Figure 3, our model performs better than the baselines over the three datasets. We should note from this figure that mview-LDA peaks at a small number of topics, however, each topic is represented by three multinomials. Moreover, it is evident from the figure that the experiment over the blog-2 dataset which measures each model's ability to generalizes to a totally unseen new blog is a harder task than generalizing to unseen posts form the same blog. However, our model still performs competitively with the SVM baseline. We believe that separating each topic into an ideology-independent part and ideology-specific part is the key behind this performance, as it is expected that the new blogs would still share much of the ideology-independent parts of the topics and hopefully would use similar (but

[4] (Lacoste-Julien et al., 2008) gave an optimization algorithm for learning the topic structure (transformation matrix), however since the code is not available, we resorted to one of the fixed splitting strategies mentioned in the paper. We tried other splits but this one gives the best results



(a) (b) (c)

Figure 3: Classification accuracy over the Bitterlemons dataset in (a) and over the two blog datasets in (b) and (c). For SVM we give the best result obtained across a wide range of the SVM's regularization parameter(not the cross-validation result).

no necessarily all) words from the ideology-specific parts of each topic when addressing this topic.

Finally, it should be noted that the bitterlemons dataset is a multi-author dataset and thus the models were tested on some authors that were not seen during training, however, two factors contributed to the good performance by all models over this dataset. The first being the larger size of each document (740 words per document as compared to 200 words per post in blog-2) and the second being the more formal writing style in the bitterlemons dataset.

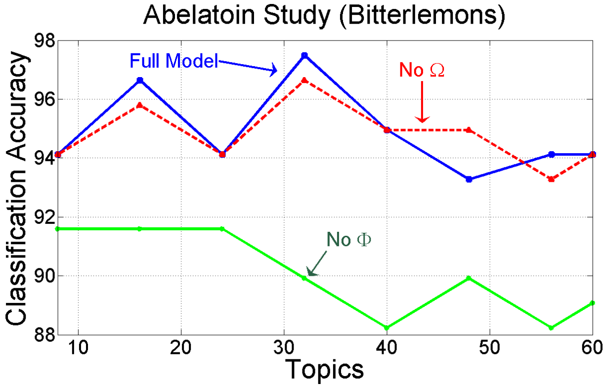


Figure 4: An Ablation study over the bitterlemons dataset.

6.3 An Ablation Study

To understand the contribution of each component of our model, we conducted an ablation study over the bitterlemons dataset. In this experiment we turned-off one feature of our model at a time and measured the classification performance. The results are shown in Figure 4. Full, refers to the full model; No-Ω refers to a model in which the ideology-specific background topic Ω is turned-off; and No-ϕ refers to a model in which the ideology-specific portions of the topics are turned-off. As evident from the figure, ϕ is more important to the model than Ω and the difference in performance between the full model and the No-ϕ model is rather significant. In fact without ϕ the model has little power to discriminate between ideologies beyond the ideology-specific background topic Ω.

6.4 Retrieval: Getting the Other Views

To evaluate the ability of our model in finding alternative views toward a given topic, we conducted the following experiment over the Bitterlemons corpus. In this corpus each document is associated with a meta-topic that highlights the issues addressed in this document like: "A possible Jordanian role", "Demography and the conflict",etc. There are a total of 148 meta-topics. These topics were not used in fitting our model but we use them in the evaluation as follows. We divided the dataset into 60% for training and 40% for testing. We trained mview-LDA over the training set, and then used the learned model to infer the latent representation of the test documents as well as their ideologies. We then used each document in the training set as a query to retrieve documents from the test set that address the same meta-topic in the query document but from the other-side's perspective. Note that we have access to the view of the query document but not the view of the test document. Moreover, the value of the meta-topic is only used to construct the ground-truth result of each query over the test set. In addition to mview-LDA, we also implemented a strong baseline using SVM+Dirichlet smoothing that we will refer to as LM. In this baseline, we build an SVM classifier over the training set, and use Dirichlet-smoothing to represent each document (in test and training set) as a multinomial-distribution over the vocabulary. Given a query document d, we rank documents in

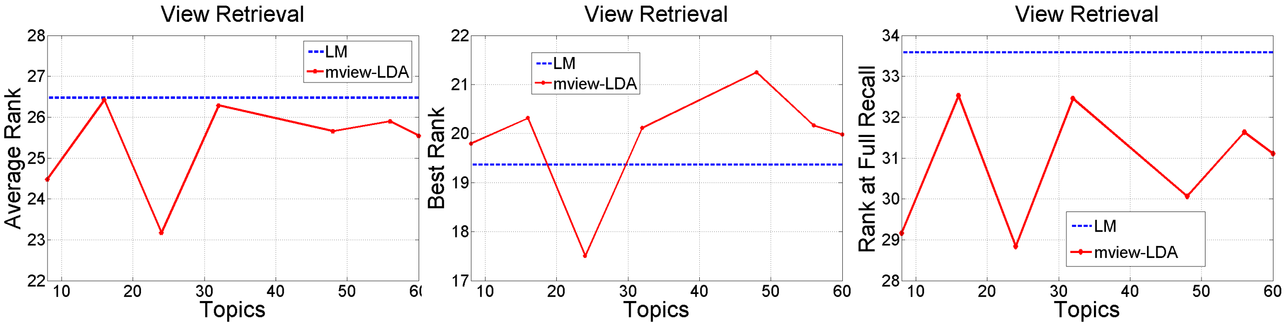


Figure 5: Evaluating the performance of the view-Retrieval task. Figure compare performance between mview-LD vs. an SVM+a smoothed language model approach using three measures: average rank, best rank and rank at full recall. (*Lower better*)

.

the test set by each model as follows:

- mview-LDA: we computed the cosine-distance between $\theta_d^{\mathrm{mv-LDA}-shared}$ and $\theta_{d'}^{\mathrm{mv-LDA-shared}}$ weighted by the probability that d' is written from a different view than v_d. The latter quantity can be computed by normalizing $P(v|d')$. Moreover, $\theta_{d,k}^{\mathrm{mv-LDA}-shared} \propto \sum_n I\big[(x_{n,1}=0) \text{ and } (x_{n,2}=1) \text{ and } (z_n=k)\big]$, and n ranges over words in document d. Intuitively, we would like $\theta_d^{\mathrm{mv-LDA}-shared}$ to reflect variation due to the topical content, but not ideological view of the document.

- LM: For a document d', we apply the SVM classifier to get $P(v|d')$, then we measure similarity by computing the cosine-distance between the smoothed multinomial-distribution of d and d'. We combine these two components as in mview-LDA.

Finally we rank documents in the test set in a descending-order and evaluate the resulting ranking using three measures: the rank at full recall (lowest rank), average rank, and best rank of the ground-truth documents as they appear in the predicted ranking. Figure 5 shows the results across a number of topics. From this figure, it is clear that our model outperforms this baseline over all measures. It should be noted that this is a very hard task since the meta-topics are very fine-grained like: Settlements revisited, The status of the settlements, Is the roadmap still relevant?,The ceasefire and the roadmap: a progress report,etc. We did not attempt to cluster these meta-topics since our goal is just to compare our model against the baseline.

7 A Semi-Supervised Extension

In this section we present and assess the efficacy of a semi-supervised extension of mview-LDA. In this setting, the model is given a set of ideologically-labeled documents and a set of unlabeled documents. One of the key advantages of using a probabilistic graphical model is the ability to deal with hidden variables in a principled way. Thus the only change needed in this case is adding a single step in the sampling algorithm to sample the ideology v of an unlabeled document as follows:

$$P(v_d = v|rest) \quad \propto \quad P(\mathbf{w_d}|v_d = v, \mathbf{z_d}, \mathbf{x_{1,d}}, \mathbf{x_{2,d}})$$

Note that the probability of the indicators $(\mathbf{x_{1,d}}, \mathbf{x_{2,d}}, \mathbf{z_d})$ do not depend on the view of the document and thus got absorbed in the normalization constant, and thus one only needs to measures the likelihood of generating the words in the document under the view v. We divide the words into three groups: $A_d = \{w_n | x_{n,1} = 1\}$ is the set of words generated from the view-background topic, $B_{d,k} = \{w_n | z_n = k, x_{n,1} = 0, x_{n,2} = 1\}$ is the set of words generated from β_k, and $C_{d,k} = \{w_n | z_n = k, x_{n,1} = 0, x_{n,2} = 0\}$ is the set of words generated from $\phi_{k,v}$. The probability of $B_{d,k}$ does not depend on the value of v and thus can be absorbed into the normalization factor. Therefore, we only need to compute the following probability:$P(A_d, C_{d,1:K}|v_d = v, rest)$=

$$\prod_k \int_{\phi_{k,v}} P(C_{d,k}|\phi_{k,v}, rest) p(\phi_{k,v}|rest) d\phi_{k,v}$$
$$\times \quad \int_\Omega P(A_d|\Omega, rest) p(\Omega|rest) d\Omega \qquad (2)$$

All the integrals in (2) reduce to the ratio of two log partition functions. For example, the product of integrals containing $C_{d,k}$ reduce to:

$$\prod_k \frac{\prod_w \Gamma\Big(C_{dkw}^{DKW,X_2=0} + C_{vkw,-d}^{VKW} + \alpha_1\Big)}{\Gamma\Big(\sum_w \big[C_{dkw}^{DKW,X_2=0} + C_{vkw,-d}^{VKW} + \alpha_1\big]\Big)} \times \frac{\Gamma\Big(\sum_w \big[C_{vkw}^{VKW} + \alpha_1\big]\Big)}{\prod_w \Gamma\Big(C_{vkw,-d}^{VKW} + \alpha_1\Big)} \tag{3}$$

Unfortunately, the above scheme does not mix well because the value of the integrals in (2) are very low for any view other than the view of the document in the current state of the sampler. This happens because of the tight coupling between v_d and the indicators $(\mathbf{x_1}, \mathbf{x_2}, \mathbf{z})$. To remedy this problem we used a Metropolis-Hasting step to sample $(v_d, \mathbf{x_1}, \mathbf{x_2}, \mathbf{z})$ jointly. We construct a set of V proposals each of which is indexed by a possible view: $q_v(\mathbf{x_1}, \mathbf{x_2}, \mathbf{z}) = P(\mathbf{x_1}, \mathbf{x_2}, \mathbf{z} | v_d = v, \mathbf{w_d})$. Since we have a collection of proposal distributions, we select one of them at random at each step. To generate a sample from $q_{v*}()$, we run a few iterations of a restricted Gibbs scan over the document d conditioned on fixing $v_d = v*$ and then take the last sample jointly with $v*$ as our proposed new state. With probability min(r,1), the new state $(v*, \mathbf{x_1}*, \mathbf{x_2}*, \mathbf{z}*)$ is accepted, otherwise the old state is retained. The acceptance ratio,r, is computed as: $r = \frac{p(\mathbf{w_d}|v*, \mathbf{x_1}*, \mathbf{x_2}*, \mathbf{z}*)}{p(\mathbf{w_d}|v, \mathbf{x_1}, \mathbf{x_2}, \mathbf{z})}$, where the non-* variables represent the current state of the sampler. It is interesting to note that the above acceptance ratio is equivalent to a likelihood ratio test. We compute the marginal probability $P(\mathbf{w_d}|..)$ using the estimated-theta method (Wallach et al., 2009).

We evaluated the semi-supervised extension using the blog-2 dataset as follows. We reveal $R\%$ of the labels in the training set; then we train mview-LDA only over the labeled portion and train the semi-supervised version (ss-mview-LDA) on both the labeled and unlabeled documents. Finally we evaluate the classification performance on the test set. We used $R = \{20, 40, 80\}$. The results are given in Table 1 which shows a decent improvement over the supervised mview-LDA.

R	mview-LDA	ss-mview-LDA
80	65.60	66.41
60	62.31	65.43
20	60.87	63.25

Table 1: Classification performance of the semi-supervised model. R is the ratio of labeled documents.

8 Discussion and Future Work

In this paper, we addressed the problem of modeling ideological perspective at a topical level. We developed a factored topic model that we called multiView-LDA or mview-LDA for short. mview-LDA factors a document collection into three set of topics: ideology-specific, topic-specific, and ideology-topic ones. We showed that the resulting representation can be used to give a bird-eyes' view to where each ideology stands with regard to mainstream topics. Moreover, we illustrated how the latent structure induced by the model can by used to perform bias-detection at the document and topic level, and retrieve documents that represent alternative views.

It is important to mention that our model induces a hierarchical structure over the topics, and thus it is interesting to contrast it with hierarchical topic models like hLDA (Blei et al., 2003) and PAM (Li and McCallum, 2006; Mimno et al., 2007). First, these models are unsupervised in nature, while our model is supervised. Second, the semantic of the hierarchical structure in our model is different than the one induced by those models since documents in our model are constrained to use a specific portion of the topic structure while in those models documents can freely sample words from any topic. Finally,in the future we plan to extend our model to perform joint modeling and summarization of idealogical discourse.

9 Acknowledgment

We thank Jacob Eisenstein, John Lafferty, Tom Mitchell, and the anonymous reviewers for their helpful comments and suggestions. This work is supported in part by grants NSF IIS- 0713379, NSF DBI-0546594 career award to EPX, ONR N000140910758, DARPA NBCH1080007, and AFOSR FA9550010247. EPX is supported by an Alfred P. Sloan Research Fellowship.

References

J. Wiebe, T. Wilson, R.Bruce, M. Bell, and M. Martin. Learning subjective language. Computational Linguistics, 30(3), 2004.

H. Yu and V. Hatzivassiloglou. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In Proceedings of EMNLP-2003

B. Pang, L. Lee, and S. Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In Proceedings of EMNLP-2002.

P. Turney and M. Littman. Measuring praise and criticism: Inference of semantic orientation from association. ACM TOIS, 21(4):315346, 2003

A. Popescu and O. Etzioni. Extracting product features and opinions from reviews. In Proceedings of HLT/EMNLP-2005, pages 339346, 2005.

T. Nasukawa and J. Yi. Sentiment analysis: Capturing favorability using natural language processing. In Proceedings of K-CAP, 2003.

M. Hu and B. Liu. Mining and summarizing customer reviews. In Proceedings of KDD, 2004.

B. Pang and L. Lee. Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval, 2(12), 1135, 2008.

S. Branavan, H. Chen, J. Eisenstein and R. Barzilay. Learning Document-Level Semantic Properties from Free-text Annotations, Proceedings of ACL, 2008.

I. Titov and R. McDonald. Modeling Online Reviews with Multi-Grain Topic Models International World Wide Web Conference (WWW), 2008.

I. Titov and R. McDonald. A Joint Model of Text and Aspect Ratings for Sentiment Summarization Association for Computational Linguistics (ACL), 2008.

Q. Mei, X. Ling, M. Wondra, H. Su, ChengXiang Zhai. Topic Sentiment Mixture: Modeling Facets and Opinions in Weblogs, Proceedings of the 16th International World Wide Web Conference (WWW), pages 171-180, 2007.

X. Ling, Q. Mei, C. Zhai, B. Schatz. Mining Multi-Faceted Overviews of Arbitrary Topics in a Text Collection, Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD' 08), pages 497-505, 2008

B. Fortuna , C. Galleguillos, N. Cristianini. Detecting the bias in media with statistical learning methods. In: Text Mining: Theory and Applications. Taylor and Francis Publisher,2008.

W. Lin, E.P. Xing, and A. Hauptmann. A Joint Topic and Perspective Model for Ideological Discourse European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD), 2008.

T. A. Van Dijk. Ideology: A multidisciplinary approach. Sage Publications, 1998.

T. Griffiths, M. Steyvers Finding scientific topics.PNAS, 101:5228-5235, 2004.

A. Gelman, J. Carlin, Hal Stern, and Donald Rubin. Bayesian Data Analysis, Chapman-Hall, 2 edition,2003.

D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. Journal of Machine Learning Research, 3:9931022, January 2003.

T. Yano, W. W. Cohen, and N. A. Smith. Predicting Response to Political Blog Posts with Topic Models. NAACL-HLT 2009, Boulder, CO, MayJune 2009

J. Eisenstein and E.P. Xing.The CMU-2008 Political Blog Corpus. CMU-ML-10-101 Technical Report, 2010.

C. Wang, D. Blei and L. Fei-Fei. Simultaneous image classification and annotation. CVPR, 2010.

D. Blei and J. McAuliffe. Supervised topic models. NIPS, 2007.

S. Lacoste-Julien, F. Sha, and M. Jordan. DiscLDA: Discriminative Learning for Dimensionality Reduction and Classification. Neural Information Processing Systems Conference (NIPS08), Vancouver, British Columbia, December 2008.

E. Riloff, J. Wiebe, and T. Wilson. Learning subjective nouns using extraction pattern bootstrapping. In Proceedings of CoNLL-2003.

D. Blei, T. Griffiths, M. Jordan, and J. Tenenbaum. Hierarchical topic models and the nested Chinese restaurant process. In Neural Information Processing Systems (NIPS)16, 2003.

D. Mimno, W. Li and A. McCallum. Mixtures of Hierarchical Topics with Pachinko Allocation. In International Conference of Machine Learning, ICML, 2007.

W. Li, and A. McCallum. Pachinko Allocation: DAG-structured Mixture Models of Topic Correlations. In International Conference of Machine Learning, ICML, 2006.

M. Paul and R. Girju. Cross-cultural Analysis of Blogs and Forums with Mixed-Collection Topic Models. EMNLP 2009.

H. Wallach, I. Murray, R. Salakhutdinov, and D. Mimno. Evaluation Methods for Topic Models. ICML 2009.

Word-based dialect identification with georeferenced rules

Yves Scherrer
LATL
Université de Genève
Genève, Switzerland
yves.scherrer@unige.ch

Owen Rambow
CCLS
Columbia University
New York, USA
rambow@ccls.columbia.edu

Abstract

We present a novel approach for (written) dialect identification based on the discriminative potential of entire words. We generate Swiss German dialect words from a Standard German lexicon with the help of hand-crafted phonetic/graphemic rules that are associated with occurrence maps extracted from a linguistic atlas created through extensive empirical fieldwork. In comparison with a character-n-gram approach to dialect identification, our model is more robust to individual spelling differences, which are frequently encountered in non-standardized dialect writing. Moreover, it covers the whole Swiss German dialect continuum, which trained models struggle to achieve due to sparsity of training data.

1 Introduction

Dialect identification (dialect ID) can be viewed as an instance of language identification (language ID) where the different languages are very closely related. Written language ID has been a popular research object in the last few decades, and relatively simple algorithms have proved to be very successful. The central question of language ID is the following: given a segment of text, which one of a predefined set of languages is this segment written in? Language identification is thus a classification problem.

Dialect identification comes in two flavors: spoken dialect ID and written dialect ID. These two tasks are rather different. Spoken dialect ID relies on speech recognition techniques which may not cope well with dialectal diversity. However, the acoustic signal is also available as input. Written dialect ID has to deal with non-standardized spellings that may occult real dialectal differences. Moreover, some phonetic distinctions cannot be expressed in orthographic writing systems and limit the input cues in comparison with spoken dialect ID.

This paper deals with written dialect ID, applied to the Swiss German dialect area. An important aspect of our model is its conception of the dialect area as a continuum without clear-cut borders. Our dialect ID model follows a bag-of-words approach based on the assumption that every dialectal word form is defined by a probability with which it may occur in each geographic area. By combining the cues of all words of a sentence, it should be possible to obtain a fairly reliable geographic localization of that sentence.

The main challenge is to create a lexicon of dialect word forms and their associated probability maps. We start with a Standard German word list and use a set of phonetic, morphological and lexical rules to obtain the Swiss German forms. These rules are manually extracted from a linguistic atlas. This linguistic atlas of Swiss German dialects is the result of decades-long empirical fieldwork.

This paper is organized as follows. We start with an overview of relevant research (Section 2) and present the characteristics of the Swiss German dialect area (Section 3). Section 4 deals with the implementation of word transformation rules and the corresponding extraction of probability maps from the linguistic atlas of German-speaking Switzerland. We present our dialect ID model in Section 5 and discuss its performance in Section 6 by relating it to a baseline n-gram model.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1151–1161,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

2 Related work

Various language identification methods have been proposed in the last three decades. Hughes et al. (2006) and Řehůřek and Kolkus (2009) provide recent overviews of different approaches. One of the simplest and most popular approaches is based on character n-gram sequences (Cavnar and Trenkle, 1994). For each language, a character n-gram language model is learned, and test segments are scored by all available language models and labeled with the best scoring language model. Related approaches involve more sophisticated learning techniques (feature-based models, SVM and other kernel-based methods).

A completely different approach relies on the identification of entire high-frequency words in the test segment (Ingle, 1980). Other models have proposed to use morpho-syntactic information.

Dialect ID has usually been studied from a speech processing point of view. For instance, Biadsy et al. (2009) classify speech material from four Arabic dialects plus Modern Standard Arabic. They first run a phone recognizer on the speech input and use the resulting transcription to build a trigram language model. Classification is done by minimizing the perplexity of the trigram models on the test segment.

An original approach to the identification of Swiss German dialects has been taken by the *Chochichästli-Orakel*.[1] By specifying the pronunciation of ten predefined words, the web site creates a probability map that shows the likelihood of these pronunciations in the Swiss German dialect area. Our model is heavily inspired by this work, but extends the set of cues to the entire lexicon.

As mentioned, the ID model is based on a large Swiss German lexicon. Its derivation from a Standard German lexicon can be viewed as a case of lexicon induction. Lexicon induction methods for closely related languages using phonetic similarity have been proposed by Mann and Yarowsky (2001) and Schafer and Yarowsky (2002), and applied to Swiss German data by Scherrer (2007).

The extraction of digital data from hand-drawn dialectological maps is a time-consuming task. Therefore, the data should be made available for different uses. Our Swiss German raw data is accessible on an interactive web page (Scherrer, 2010), and we have proposed ideas for reusing this data for machine translation and dialect parsing (Scherrer and Rambow, 2010). An overview of digital dialectological maps for other languages is available on `http://www.ericwheeler.ca/atlaslist`.

3 Swiss German dialects

The German-speaking area of Switzerland encompasses the Northeastern two thirds of the Swiss territory, and about two thirds of the Swiss population define (any variety of) German as their first language.

In German-speaking Switzerland, dialects are used in speech, while Standard German is used nearly exclusively in written contexts (diglossia). It follows that all (adult) Swiss Germans are bidialectal: they master their local dialect and Standard German. In addition, they usually have no difficulties understanding Swiss German dialects other than their own.

Despite the preference for spoken dialect use, written dialect data has been produced in the form of dialect literature and transcriptions of speech recordings made for scientific purposes. More recently, written dialect has been used in electronic media like blogs, SMS, e-mail and chatrooms. The Alemannic Wikipedia contains about 6000 articles, among which many are written in a Swiss German dialect.[2] However, all this data is very heterogeneous in terms of the dialects used, spelling conventions and genre.

4 Georeferenced word transformation rules

The key component of the proposed dialect ID model is an automatically generated list of Swiss German word forms, each of which is associated with a map that specifies its likelihood of occurrence over German-speaking Switzerland. This word list is generated with the help of a set of transformation rules, taking a list of Standard German words as a starting point. In this section, we present the different types of rules and how they can be extracted from a dialectological atlas.

[1] `http://dialects.from.ch`

[2] `http://als.wikipedia.org`; besides Swiss German, the Alemannic dialect group encompasses Alsatian, South-West German Alemannic and Vorarlberg dialects of Austria.

4.1 Orthography

Our system generates written dialect words according to the Dieth spelling conventions without diacritics (Dieth, 1986).[3] These are characterized by a transparent grapheme-phone correspondence and are widely used by dialect writers. However, they are by no means enforced or even taught.

This lack of standardization is problematic for dialect ID. We have noted two major types of deviations from the Dieth spelling conventions in our data. First, Standard German orthography may unduly influence dialect spelling. For example, *spiele* is modelled after Standard German *spielen* 'to play', although the vowel is a short monophthong in Swiss German and should thus be written *spile* (*ie* represents a diphthong in Dieth spelling). Second, dialect writers do not always distinguish short and long vowels, while the Dieth conventions always use letter doubling to indicate vowel lengthening. Future work will incorporate these fluctuations directly into the dialect ID model.

Because of our focus on written dialect, the following discussion will be based on written representations, but IPA equivalents are added for convenience.

4.2 Phonetic rules

Our work is based on the assumption that many words show predictable phonetic differences between Standard German and the different Swiss German dialects. Hence, in many cases, it is not necessary to explicitly model word-to-word correspondences, but a set of phonetic rules suffices to correctly transform words.

For example, the word-final sequence *nd* [nd̥] (as in Standard German *Hund* 'dog'[4]) is maintained in most Swiss German dialects. However, it has to be transformed to *ng* [ŋ] in Berne dialect, to *nn* [n] in Fribourg dialect, and to *nt* [nt] in Valais and Uri dialects.

This phenomenon is captured in our system by four transformation rules *nd* → *nd*, *nd* → *ng*, *nd* → *nn* and *nd* → *nt*. Each rule is *georeferenced*, i.e. linked to a probability map that specifies its validity in every geographic point. These four *rules* capture one single linguistic *phenomenon*: their left-hand side is the same, and they are geographically complementary.

Some rules apply uniformly to all Swiss German dialects (e.g. the transformation *st* [st] → *scht* [ʃt]). These rules do not immediately contribute to the dialect identification task, but they help to obtain correct Swiss German forms that contain other phonemes with better localization potential.

More information about the creation of the probability maps is given in Sections 4.5 and 4.6.

4.3 Lexical rules

Some differences at the word level cannot be accounted for by pure phonetic alternations. One reason are idiosyncrasies in the phonetic evolution of high frequency words (e.g. Standard German *und* 'and' is reduced to *u* in Bern dialect, where the phonetic rules would rather suggest **ung*). Another reason is the use of different lexemes altogether (e.g. Standard German *immer* 'always' corresponds to *geng*, *immer*, or *all*, depending on the dialect). We currently use lexical rules mainly for function words and irregular verb stems.

4.4 Morphological rules

The transformation process from inflected Standard German word forms to inflected Swiss German word forms is done in two steps. First, the word stem is adapted with phonetic or lexical rules, and then, the affixes are generated according to the morphological features of the word.

Inflection markers also provide dialect discrimination potential. For example, the verbal plural suffixes offer a surprisingly rich (and diachronically stable) interdialectal variation pattern.

4.5 The linguistic atlas SDS

One of the largest research projects in Swiss German dialectology has been the elaboration of the *Sprachatlas der deutschen Schweiz* (SDS), a linguistic atlas that covers phonetic, morphological and lexical differences of Swiss German dialects. Data collection and publication were carried out between 1939 and 1997 (Hotzenköcherle et al., 1962-1997). Linguistic data were collected in about 600 villages (*inquiry points*) of German-speaking Switzerland, and

[3] Of course, these spelling conventions make use of umlauts like in Standard German. There is another variant of the Dieth conventions that uses additional diacritics for finer-grained phonetic distinctions.

[4] Standard German *nd* is always pronounced [nt] following a general final devoicing rule; we neglect that artifact as we rely only on graphemic representations.

resulted in about 1500 published maps (see Figure 1 for an example).

Each map represents a linguistic phenomenon that potentially yields a set of transformation rules. For our experiments, we selected a subset of the maps according to the perceived importance of the described phenomena. There is no one-to-one correspondence between maps and implemented phenomena, for several reasons. First, some SDS maps represent information that is best analyzed as several distinct phenomena. Second, a set of maps may illustrate the same phenomenon with different words and slightly different geographic distributions. Third, some maps describe (especially lexical) phenomena that are becoming obsolete and that we chose to omit.

As a result, our rule base contains about 300 phonetic rules covering 130 phenomena, 540 lexical rules covering 250 phenomena and 130 morphological rules covering 60 phenomena. We believe this coverage to be sufficient for the dialect ID task.

4.6 Map digitization and interpolation

Recall the *nd*-example used to illustrate the phonetic rules above. Figure 1 shows a reproduction of the original, hand-drawn SDS map related to this phenomenon. Different symbols represent different phonetic variants of the phenomenon.[5] We will use this example in this section to explain the preprocessing steps involved in the creation of georeferenced rules.

In a first preprocessing step, the hand-drawn map is digitized manually with the help of a geographical information system. The result is shown in Figure 2. To speed up this process, variants that are used in less than ten inquiry points are omitted. (Many of these small-scale variants likely have disappeared since the data collection in the 1940s.) We also collapse minor phonetic variants which cannot be distinguished in the Dieth spelling system.

The SDS maps, hand-drawn or digitized, are point maps. They only cover the inquiry points, but do not provide information about the variants used in other locations. Therefore, a further preprocessing step interpolates the digitized point maps to obtain surface maps. We follow Rumpf et al. (2009) to create kernel density estimators for each variant. This method is

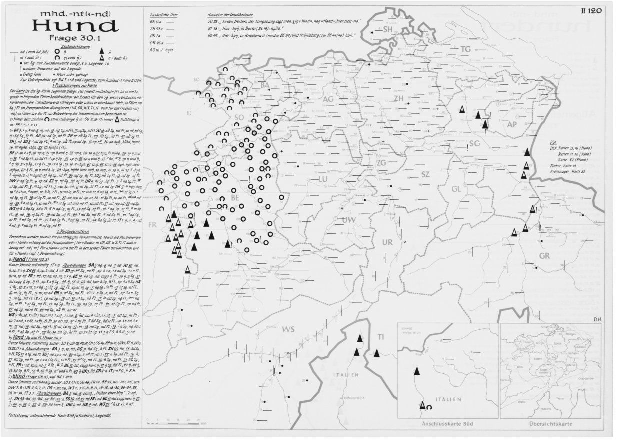


Figure 1: Original SDS map for the transformation of word-final *-nd*. The map contains four major linguistic variants, symbolized by horizontal lines (*-nd*), vertical lines (*-nt*), circles (*-ng*), and triangles (*-nn*) respectively. Minor linguistic variants are symbolized by different types of circles and triangles.

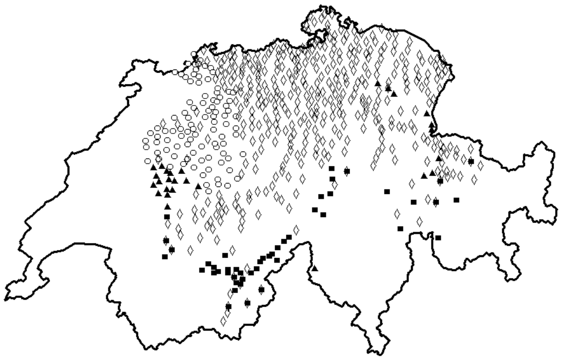


Figure 2: Digitized equivalent of the map in Figure 1.

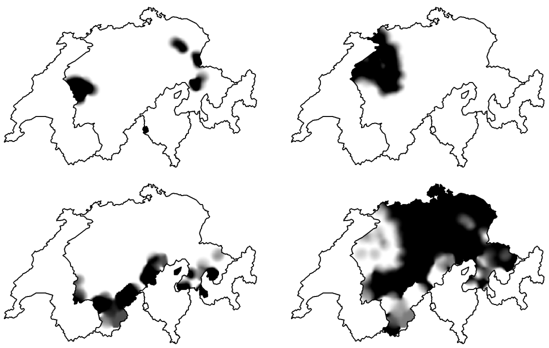


Figure 3: Interpolated surface maps for the variants *-nn* (upper left), *-ng* (upper right), *-nt* (lower left) and *-nd* (lower right). Black areas represent a probability of 1, white areas a probability of 0.

[5] We define a *variant* simply as a string that may occur on the right-hand side of a transformation rule.

less sensitive to outliers than simpler linear interpolation methods.[6] The resulting surface maps are then normalized such that at each point of the surface, the weights of all variants sum up to 1. These normalized weights can be interpreted as conditional probabilities of the corresponding transfer rule: $p(r \mid t)$, where r is the rule and t is the geographic location (represented as a pair of longitude and latitude coordinates) situated in German-speaking Switzerland. (We call the set of all points in German-speaking Switzerland *GSS*.) Figure 3 shows the resulting surface maps for each variant. Surface maps are generated with a resolution of one point per square kilometer.

As mentioned above, rules with a common left-hand side are grouped into phenomena, such that at any given point $t \in GSS$, the probabilities of all rules r describing a phenomenon Ph sum up to 1:

$$\underset{t \in GSS}{\forall} \sum_{r \in Ph} p(r \mid t) = 1$$

5 The model

The dialect ID system consists of a Swiss German lexicon that associates word forms with their geographical extension (Section 5.1), and of a testing procedure that splits a sentence into words, looks up their geographical extensions in the lexicon, and condenses the word-level maps into a sentence-level map (Sections 5.2 to 5.4).

5.1 Creating a Swiss German lexicon

The Swiss German word form lexicon is created with the help of the georeferenced transfer rules presented above. These rules require a lemmatized, POS-tagged and morphologically disambiguated Standard German word as an input and generate a set of dialect word/map tuples: each resulting dialect word is associated with a probability map that specifies its likelihood in each geographic point.

To obtain a Standard German word list, we extracted all leaf nodes of the TIGER treebank (Brants et al., 2002), which are lemmatized and morphologically annotated. These data also allowed us to obtain word frequency counts. We discarded words with one single occurrence in the TIGER treebank, as well as forms that contained the genitive case or preterite tense attribute (the corresponding grammatical categories do not exist in Swiss German dialects).

The transfer rules are then applied sequentially on each word of this list. The notation $w_0 \overset{*}{\rightarrow} w_n$ represents an iterative derivation leading from a Standard German word w_0 to a dialectal word form w_n by the application of n transfer rules of the type $w_i \rightarrow w_{i+1}$. The probability of a derivation corresponds to the joint probability of the rules it consists of. Hence, the probability map of a derivation is defined as the **pointwise product** of all rule maps it consists of:

$$\underset{t \in GSS}{\forall} p(w_0 \overset{*}{\rightarrow} w_n \mid t) = \prod_{k=0}^{n-1} p(w_i \rightarrow w_{i+1} \mid t)$$

Note that in dialectological transition zones, there may be several valid outcomes for a given w_0.

The Standard German word list extracted from TIGER contains about 36,000 entries. The derived Swiss German word list contains 560,000 word forms, each of which is associated with a map that specifies its regional distribution.[7] Note that proper nouns and words tagged as “foreign material” were not transformed. Derivations that did not obtain a probability higher than 0.1 anywhere (because of geographically incompatible transformations) were discarded.

5.2 Word lookup and dialect identification

At test time, the goal is to compute a probability map for a text segment of unknown origin.[8] As a preprocessing step, the segment is tokenized, punctuation markers are removed and all words are converted to lower case.

The identification process can be broken down in three levels:

1. The probability map of a text segment depends on the probability maps of the words contained in the segment.

2. The probability map of a word depends on the probability maps of the derivations that yield the word.

[6] A comparison of different interpolation methods will be the object of future work.

[7] Technically, we do not store the probability map, but the sequence of rule variants involved in the derivation. The probability map is restored from this rule sequence at test time.

[8] The model does not require the material to be syntactically well-formed. Although we use complete sentences to test the system, any sequence of words is accepted.

3. The probability map of a derivation depends on the probability maps of the rules it consists of.

In practice, every word of a given text segment is looked up in the lexicon. If this lookup does not succeed (either because its Standard German equivalent did not appear in the TIGER treebank, or because the rule base lacked a relevant rule), the word is skipped. Otherwise, the lookup yields *m* derivations from *m* different Standard German words.[9] The lexicon already contains the probability maps of the derivations (see 5.1), so that the third level does not need to be discussed here. Let us thus explain the first two levels in more detail, in reverse order.

5.3 Computing the probability map for a word

A dialectal word form may originate in different Standard German words. For example, the three derivations *sind [VAFIN]* $\xrightarrow{*}$ *si* (valid only in Western dialects), *sein [PPOSAT]* $\xrightarrow{*}$ *si* (in Western and Central dialects), and *sie [PPER]* $\xrightarrow{*}$ *si* (in the majority of Swiss German dialects) all lead to the same dialectal form *si*.

Our system does not take the syntactic context into account and therefore cannot determine which derivation is the correct one. We approximate by choosing the most probable one in each geographic location. The probability map of a Swiss German word *w* is thus defined as the **pointwise maximum**[10] of all derivations leading to *w*, starting with different Standard German words $w_0^{(j)}$:

$$\underset{t \in GSS}{\forall} \; p(w \mid t) = \max_j p(w_0^{(j)} \xrightarrow{*} w \mid t)$$

This formula does not take into account the relative frequency of the different derivations of a word. This may lead to unintuitive results. Consider the two derivations *der [ART]* $\xrightarrow{*}$ *dr* (valid only in Western dialects) and *Dr. [NN]* $\xrightarrow{*}$ *dr* (valid in all dialects). The occurrence of the article *dr* in a dialect text is a good indicator for Western Swiss dialects, but it is completely masked by the potential presence of the abreviation *Dr.* in all dialects. We can avoid this by weighting the derivations by the **word frequency** of w_0: the article *der* is much more frequent than the abreviation *Dr.* and is thus given more weight in the identification task. This weighting can be justified on dialectological grounds: frequently used words tend to show higher interdialectal variation than rare words.

Another assumption in the above formula is that each derivation has the same **discriminative potential**. Again, this is not true: a derivation that is valid in only 10% of the Swiss German dialect area is much more informative than a derivation that is valid in 95% of the dialect area. Therefore, we propose to weight each derivation by the proportional size of its validity area. The discriminative potential of a derivation *d* is defined as follows:[11]

$$DP(d) = 1 - \frac{\sum_{t \in GSS} p(d \mid t)}{|GSS|}$$

The experiments in Section 6 will show the relative impact of these two weighting techniques and of the combination of both with respect to the unweighted map computation.

5.4 Computing the probability map for a segment

The probability of a text segment *s* can be defined as the joint probability of all words *w* contained in the segment. Again, we compute the **pointwise product** of all word maps. In contrast to 5.1, we performed some smoothing in order to prevent erroneous word derivations from completely zeroing out the probabilities. We assumed a minimum word probability of $\phi = 0.1$ for all words in all geographic points:

$$\underset{t \in GSS}{\forall} \; p(s \mid t) = \prod_{w \in s} \max(\phi, p(w \mid t))$$

Erroneous derivations were mainly due to non-implemented lexical exceptions.

6 Experiments and results

6.1 Data

In order to evaluate our model, we need texts annotated with their gold dialect. We have chosen to use the Alemannic Wikipedia as a main data source.

[9] Theoretically, two derivations can originate at the same Standard German word and yield the same Swiss German word, but nevertheless use different rules. Our system handles such cases as well, but we are not aware of such cases occurring with the current rule base.

[10] Note that these derivations are alternatives and not joint events. This is thus not a joint probability.

[11] d is a notational abreviation for $w_0 \xrightarrow{*} w_n$.

Wikipedia name	Abbr.	Pop.	Surface
Baseldytsch	BA	8%	1%
Bärndütsch	BE	17%	13%
Seislertütsch	FR	2%	1%
Ostschwizertütsch	OS	14%	8%
Wallisertiitsch	WS	2%	7%
Züritüütsch	ZH	22%	4%

Table 1: The six dialect regions selected for our tests, with their annotation on Wikipedia and our abreviation. We also show the percentage of the German-speaking population living in the regions, and the percentage of the surface of the region relative to the entire country.

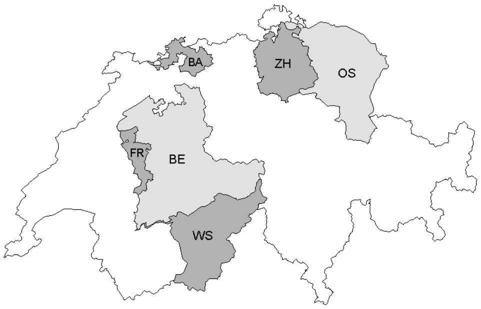


Figure 4: The localization of the six dialect regions used in our study.

The Alemannic Wikipedia allows authors to write articles in any dialect, and to annotate the articles with their dialect. Eight dialect categories contained more than 10 articles; we selected six dialects for our experiments (see Table 1 and Figure 4).

We compiled a test set consisting of 291 sentences, distributed across the six dialects according to their population size. The sentences were taken from different articles. In addition, we created a development set consisting of 550 sentences (100 per dialect, except FR, where only 50 sentences were available). This development set was also used to train the baseline model discussed in section 6.2.

In order to test the robustness of our model, we collected a second set of texts from various web sites other than Wikipedia. The gold dialect of these texts could be identified through metadata.[12] This information was checked for plausibility by the first author. The Web data set contains 144 sentences (again distributed according to population size) and is thus roughly half the size of the Wikipedia test set.

	Wikipedia			Web		
Dialect	P	R	F	P	R	F
BA	34	61	44	27	61	37
BE	78	51	61	51	47	49
FR	28	71	40	10	33	15
OS	63	64	64	50	38	43
WS	58	100	74	14	33	20
ZH	77	62	69	77	41	53
W. Avg.			62			46

Table 2: Performances of the 5-gram model on Wikipedia test data (left) and Web test data (right). The columns refer to precision, recall and F-measure respectively. The average is weighted by the relative population sizes of the dialect regions.

The Wikipedia data contains an average of 17.8 words per sentence, while the Web data shows 14.9 words per sentence on average.

6.2 Baseline: N-gram model

To compare our dialect ID model, we created a baseline system that uses a character-n-gram approach. This approach is fairly common for language ID and has also been successfully applied to dialect ID (Biadsy et al., 2009). However, it requires a certain amount of training data that may not be available for specific dialects, and it is uncertain how it performs with very similar dialects.

We trained 2-gram to 6-gram models for each dialect with the SRILM toolkit (Stolcke, 2002), using the Wikipedia development corpus. We scored each sentence of the Wikipedia test set with each dialect model. The predicted dialect was the one which obtained the lowest perplexity.[13]

The 5-gram model obtained the best overall performance, and results on the Wikipedia test set were surprisingly good (see Table 2, leftmost columns).[14] Note that in practice, 100% accuracy is not always achievable; a sentence may not contain a sufficient localization potential to assign it unambiguously to one dialect.

[12] We mainly chose websites of local sports and music clubs, whose localization allowed to determine the dialect of their content.

[13] We assume that all test sentences are written in one of the six dialects.

[14] All results represent percentage points. We omit decimal places as all values are based on 100 or less data points. We did not perform statistical significance tests on our data.

However, we suspect that these results are due to overfitting. It turns out that the number of Swiss German Wikipedia authors is very low (typically, one or two active writers per dialect), and that every author uses distinctive spelling conventions and writes about specific subjects. For instance, most ZH articles are about Swiss politicians, while many OS articles deal with religion and mysticism. Our hypothesis is thus that the n-gram model learns to recognize a specific author and/or topic rather than a dialect. This hypothesis is confirmed on the Web data set: the performances drop by 15 percentage points or more (same table, rightmost columns; the performance drops are similar for $n = [2..6]$).

In all our evaluations, the average F-measures for the different dialects are weighted according to the relative population sizes of the dialect regions because the size of the test corpus is proportional to population size (see Section 6.1).[15]

We acknowledge that a training corpus of only 100 sentences per dialect provides limited insight into the performance of the n-gram approach. We were able to double the training corpus size with additional Wikipedia sentences. With this extended corpus, the 4-gram model performed better than the 5-gram model. It yielded a weighted average F-measure of 79% on Wikipedia test data, but only 43% on Web data. The additional increase on Wikipedia data (+17% absolute with respect to the small training set), together with the decrease on Web data (−3% absolute) confirms our hypothesis of overfitting. An ideal training corpus should thus contain data from several sources per dialect.

To sum up, n-gram models can yield good performance even with similar dialects, but require large amounts of training data from different sources to achieve robust results. For many small-scale dialects, such data may not be available.

6.3 Our model

The n-gram system presented above has no geographic knowledge whatsoever; it just consists of six distinct language models that could be located anywhere. In contrast, our model yields probability maps of German-speaking Switzerland. In order to evaluate its performance, we thus had to determine the geographic localization of the six dialect regions defined by the Wikipedia authors (see Table 1). We defined the regions according to the respective canton boundaries and to the German-French language border in the case of bilingual cantons. The result of this mapping is shown in Figure 4.

The predicted dialect region of a sentence s is defined as the region in which the most probable point has a higher value than the most probable point in any other region:

$$Region(s) = \arg\max_{Region} \left(\max_{t \in Region} p(s \mid t) \right)$$

Experiments were carried out for the four combinations of the two derivation-weighting techniques presented in Section 5.3 and for the two test sets (Wikipedia and Web). Results are displayed in Tables 3 to 6. The majority of FR sentences were misclassified as BE, which reflects the geographic and linguistic proximity of these regions.

The tables show that frequency weighting helps on both corpora: the discriminative potential only slightly improves performance on the web corpus. Crucially, the two techniques are additive, so in combination, they yield the best overall results. In comparison with the baseline model, there is a performance drop of about 16 percent absolute on Wikipedia data. In contrast, our model is very robust and outperforms the baseline model on the Web test set by about 7 percent absolute.

These results seem to confirm what we suggested above: that the n-gram model overfitted on the small Wikipedia training corpus. Nevertheless, it is still surprising that our model has a lower performance on Wikipedia than on Web data. The reason for this discrepancy probably lies in the spelling conventions assumed in the transformation rules: it seems that Web writers are closer to these (implicit) spelling conventions than Wikipedia authors. This may be explained by the fact that many Wikipedia articles are translations of existing Standard German articles, and that some words are not completely adapted to their dialectal form. Another reason could be that Wikipedia articles use a proportionally larger amount of proper nouns and low-frequency words which can-

[15] Roughly, this weighting can be viewed as a prior (the probability of the text being constant):

$$p(dialect \mid text) = p(text \mid dialect) * p(dialect)$$

	Wikipedia			Web		
Dialect	P	R	F	P	R	F
BA	41	19	26	80	22	35
BE	42	62	50	48	76	59
FR	0	0	0	17	33	22
OS	36	41	38	45	41	43
WS	3	14	5	8	33	13
ZH	65	33	44	62	37	46
W. Avg.			40			46

Table 3: Performances of the word-based model using unweighted derivation maps.

	Wikipedia			Web		
Dialect	P	R	F	P	R	F
BA	50	33	40	57	22	32
BE	47	60	53	60	79	68
FR	0	0	0	0	0	0
OS	29	31	30	46	50	48
WS	11	29	15	17	33	22
ZH	60	47	53	65	53	58
W. Avg.			44			**53**

Table 4: Performances of the word-based model using derivation maps weighted by word frequency.

	Wikipedia			Web		
Dialect	P	R	F	P	R	F
BA	34	31	32	38	17	23
BE	46	47	47	54	76	63
FR	11	14	13	20	33	25
OS	34	50	40	53	59	56
WS	5	14	7	0	0	0
ZH	47	27	34	75	43	55
W. Avg.			37			51

Table 5: Performances of the word-based model using derivation maps weighted by their discriminative potential.

	Wikipedia			Web		
Dialect	P	R	F	P	R	F
BA	46	28	35	33	11	17
BE	47	62	54	58	84	69
FR	0	0	0	20	33	25
OS	35	31	33	47	47	47
WS	8	29	13	14	33	20
ZH	63	53	58	66	51	58
W. Avg.			**46**			52

Table 6: Performances using derivation maps weighted by word frequency and discriminative potential.

not be found in the lexicon and which therefore reduce the localization potential of a sentence.

However, one should note that the word-based dialect ID model is not limited on the six dialect regions used for evaluation here. It can be used with any size and number of dialect regions of German-speaking Switzerland. This contrasts with the n-gram model which has to be trained specifically on every dialect region; in this case, the Swiss German Wikipedia only contains two additional dialect regions with an equivalent amount of data.

6.4 Variations

In the previous section, we have defined the predicted dialect region as the one in which the most probable point (**maximum**) has a higher probability than the most probable point of any other region. The results suggest that this metric penalizes small regions (BA, FR, ZH). In these cases, it is likely that the most probable point is slightly outside the region, but that the largest part of the probability mass is still inside the correct region. Therefore, we tested another approach: we defined the predicted dialect region as the one in which the **average** probability is higher than the average probability in any other region:

$$Region(s) = \arg\max_{Region} \left(\frac{\sum_{t \in Region} p(s \mid t)}{|Region|} \right)$$

This metric effectively boosts the performance on the smaller regions, but comes at a cost for larger regions (Table 7). We also combined the two metrics by using the **maximum** metric for the three larger regions and the **average** metric for the three smaller ones (the cutoff lies at 5% of the Swiss territory). This combined metric further improves the performance of our system while relying on an objective measure of region surface.

We believe that region surface as such is not so crucial for the metrics discussed above, but rather serves as a proxy for linguistic heterogeneity. Geographically large regions like BE tend to have internal dialect variation, and averaging over all dialects in the region leads to low figures. In contrast, small regions show a quite homogeneous dialect landscape that may protrude over adjacent regions. In this case, the probability peak is less relevant than the average probability in the entire region. Future work will attempt to come up with more fine-grained measures of

	Wikipedia			Web		
Dialect	Max	Avg	Cmb	Max	Avg	Cmb
BA	35	32	32	17	43	43
BE	54	39	54	69	54	69
FR	0	7	7	25	11	11
OS	33	23	33	47	49	47
WS	13	13	13	20	31	20
ZH	58	60	60	58	68	68
W. Avg.	46	40	**47**	52	55	**58**

Table 7: Comparison of different evaluation metrics. All values refer to F-measures obtained with frequency and discriminative potential-weighted derivation maps. Max refers to the Maximum metric as used in Table 6. Avg refers to the average metric, and Cmb is the combination of both metrics depending on region surfaces. The underlined values in the Avg and Max columns represent those used for the Cmb metric.

linguistic heterogeneity in order to test these claims.

7 Future work

In our experiments, the word-based dialect identification model skipped about one third of all words (34% on the Wikipedia test set, 39% on the Web test set) because they could not be found in the lexicon. While our model does not require complete lexical coverage, this figure shows that the system can be improved. We see two main possibilities of improvement. First, the rule base can be extended to better account for lexical exceptions, orthographic variation and irregular morphology. Second, a mixed approach could combine the benefits of the word-based model with the n-gram model. This would require a larger, more heterogeneous set of training material for the latter in order to avoid overfitting. Additional training data could be extracted from the web and automatically annotated with the current model in a semi-supervised approach.

In the evaluation presented above, the task consisted of identifying the dialect of single sentences. However, one often has access to longer text segments, which makes our evaluation setup harder than necessary. This is especially important in situations where a single sentence may not always contain enough discriminative material to assign it to a unique dialect. Testing our dialect identification system on the paragraph or document level could thus provide more realistic results.

8 Conclusion

In this paper, we have compared two empirical methods for the task of dialect identification. The n-gram method is based on the approach most commonly used in NLP: it is a supervised machine learning approach where training data of the type we need to process is annotated with the desired outcome of the processing.

Our second approach – the main contribution of this paper – is quite different. The empirical component consists in a collection of data (the SDS atlas) which is not of the type we want to process, but rather embodies some features of the data we ultimately want to process. We therefore analyze this data in order to extract empirically grounded knowledge for more general use (the creation of the georeferenced rules), and then use this knowledge to perform the dialect ID task in conjunction with an unrelated data source (the Standard German corpus).

Our choice of method was of course related to the fact that few corpora, annotated or not, were available for our task. But beyond this constraint, we think it may be well worthwhile for NLP tasks in general to move away from a narrow machine learning paradigm (supervised or not) and to consider a broader set of empirical resources, sometimes requiring methods which are quite different from the prevalent ones.

Acknowledgements

Part of this work was carried out during the first author's stay at Columbia University, New York, funded by the Swiss National Science Foundation (grant PBGEP1-125929).

References

Fadi Biadsy, Julia Hirschberg, and Nizar Habash. 2009. Spoken Arabic dialect identification using phonotactic modeling. In *EACL 2009 Workshop on Computational Approaches to Semitic Languages*, Athens.

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.

W. B. Cavnar and J. M. Trenkle. 1994. N-gram based text categorization. In *Proceedings of SDAIR'94*, Las Vegas.

Eugen Dieth. 1986. *Schwyzertütschi Dialäktschrift*. Sauerländer, Aarau, 2nd edition.

Rudolf Hotzenköcherle, Robert Schläpfer, Rudolf Trüb, and Paul Zinsli, editors. 1962-1997. *Sprachatlas der deutschen Schweiz*. Francke, Berne.

Baden Hughes, Timothy Baldwin, Steven Bird, Jeremy Nicholson, and Andrew MacKinlay. 2006. Reconsidering language identification for written language resources. In *Proceedings of LREC'06*, Genoa.

N. Ingle. 1980. A language identification table. *Technical Translation International*.

Gideon S. Mann and David Yarowsky. 2001. Multipath translation lexicon induction via bridge languages. In *Proceedings of NAACL'01*, Pittsburgh.

Radim Řehůřek and Milan Kolkus. 2009. Language identification on the web: Extending the dictionary method. In *Computational Linguistics and Intelligent Text Processing – Proceedings of CICLing 2009*, pages 357–368, Mexico. Springer.

Jonas Rumpf, Simon Pickl, Stephan Elspaß, Werner König, and Volker Schmidt. 2009. Structural analysis of dialect maps using methods from spatial statistics. *Zeitschrift für Dialektologie und Linguistik*, 76(3).

Charles Schafer and David Yarowsky. 2002. Inducing translation lexicons via diverse similarity measures and bridge languages. In *Proceedings of CoNLL'02*, pages 146–152, Taipei.

Yves Scherrer and Owen Rambow. 2010. Natural language processing for the Swiss German dialect area. In *Proceedings of KONVENS'10*, Saarbrücken.

Yves Scherrer. 2007. Adaptive string distance measures for bilingual dialect lexicon induction. In *Proceedings of ACL'07, Student Research Workshop*, pages 55–60, Prague.

Yves Scherrer. 2010. Des cartes dialectologiques numérisées pour le TALN. In *Proceedings of TALN'10*, Montréal.

Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proceedings of ICSLP'02*, pages 901–904, Denver.

Measuring Distributional Similarity in Context

Georgiana Dinu
Department of Computational Linguistics
Saarland University
Saarbrücken, Germany
dinu@coli.uni-sb.de

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

Abstract

The computation of meaning similarity as operationalized by vector-based models has found widespread use in many tasks ranging from the acquisition of synonyms and paraphrases to word sense disambiguation and textual entailment. Vector-based models are typically directed at representing words in isolation and thus best suited for measuring similarity out of context. In his paper we propose a probabilistic framework for measuring similarity in context. Central to our approach is the intuition that word meaning is represented as a probability distribution over a set of latent senses and is modulated by context. Experimental results on lexical substitution and word similarity show that our algorithm outperforms previously proposed models.

1 Introduction

The computation of meaning similarity as operationalized by vector-based models has found widespread use in many tasks within natural language processing (NLP). These range from the acquisition of synonyms (Grefenstette, 1994; Lin, 1998) and paraphrases (Lin and Pantel, 2001) to word sense disambiguation (Schuetze, 1998), textual entailment (Clarke, 2009), and notably information retrieval (Salton et al., 1975).

The popularity of vector-based models lies in their unsupervised nature and ease of computation. In their simplest incarnation, these models represent the meaning of each word as a point in a high-dimensional space, where each component corresponds to some co-occurring contextual element (Landauer and Dumais, 1997; McDonald, 2000; Lund and Burgess, 1996). The advantage of taking such a geometric approach is that the similarity of word meanings can be easily quantified by measuring their distance in the vector space, or the cosine of the angle between them.

Vector-based models do not explicitly identify the different senses of words and consequently represent their meaning invariably (i.e., irrespective of co-occurring context). Consider for example the adjective *heavy* which we may associate with the general meaning of “dense” or “massive”. However, when attested in context, *heavy* may refer to an overweight person (e.g., *She is short and heavy but she has a heart of gold.*) or an excessive cannabis user (e.g., *Some heavy users develop a psychological dependence on cannabis.*).

Recent work addresses this issue *indirectly* with the development of specialized models that represent word meaning in context (Mitchell and Lapata, 2008; Erk and Padó, 2008; Thater et al., 2009). These methods first extract typical co-occurrence vectors representing a *mixture* of senses and then use vector operations to either obtain contextualized representations of a target word (Erk and Padó, 2008) or a representation for a set of words (Mitchell and Lapata, 2009).

In this paper we propose a probabilistic framework for representing word meaning and measuring similarity in context. We model the meaning of isolated words as a probability distribution over a set of latent senses. This distribution reflects the *a priori*, out-of-context likelihood of each sense. Because sense ambiguity is taken into account *directly* in the

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1162–1172,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

vector construction process, contextualized meaning can be modeled naturally as a change in the original sense distribution. We evaluate our approach on word similarity (Finkelstein et al., 2002) and lexical substitution (McCarthy and Navigli, 2007) and show improvements over competitive baselines.

In the remainder of this paper we give a brief overview of related work, emphasizing vector-based approaches that compute word meaning in context (Section 2). Next, we present our probabilistic framework and different instantiations thereof (Sections 3 and 4). Finally, we discuss our experimental results (Sections 5 and 6) and conclude the paper with future work.

2 Related work

Vector composition methods construct representations that go beyond individual words (e.g., for phrases or sentences) and thus by default obtain word meanings in context. Mitchell and Lapata (2008) investigate several vector composition operations for representing short sentences (consisting of intransitive verbs and their subjects). They show that models performing point-wise multiplication of component vectors outperform earlier proposals based on vector addition (Landauer and Dumais, 1997; Kintsch, 2001). They argue that multiplication approximates the intersection of the meaning of two vectors, whereas addition their union. Mitchell and Lapata (2009) further show that their models yield improvements in language modeling.

Erk and Padó (2008) employ selectional preferences to contextualize occurrences of target words. For example, the meaning of a verb in the presence of its object is modeled as the multiplication of the verb's vector with the vector capturing the inverse selectional preferences of the object; the latter are computed as the centroid of the verbs that occur with this object. Thater et al. (2009) improve on this model by representing verbs in a second order space, while the representation for objects remains first order. The meaning of a verb boils down to restricting its vector to the features active in the argument noun (i.e., dimensions with value larger than zero).

More recently, Reisinger and Mooney (2010) present a method that uses clustering to produce multiple sense-specific vectors for each word. Specifically, a word's contexts are clustered to produce groups of similar context vectors. An average prototype vector is then computed separately for each cluster, producing a set of vectors for each word. These cluster vectors can be used to determine the semantic similarity of both isolated words and words in context. In the second case, the distance between prototypes is weighted by the probability that the context belongs to the prototype's cluster. Erk and Padó (2010) propose an exemplar-based model for capturing word meaning in context. In contrast to the prototype-based approach, no clustering takes place, it is assumed that there are as many senses as there are instances. The meaning of a word in context is the set of exemplars most similar to it.

Unlike Reisinger and Mooney (2010) and Erk and Padó (2010) our model is probabilistic (we represent word meaning as a distribution over a set of latent senses), which makes it easy to integrate and combine with other systems via mixture or product models. More importantly, our approach is conceptually simpler as we use a single vector representation for isolated words as well as for words in context. A word's different meanings are simply modeled as changes in its sense distribution. We should also point out that our approach is not tied to a specific sense induction method and can be used with different variants of vector-space models.

3 Meaning Representation in Context

In this section we first describe how we represent the meaning of individual words and then move on to discuss our model of inducing meaning representations in context.

Observed Representations Most vector space models in the literature perform computations on a co-occurrence matrix where each row represents a *target* word, each column a document or another neighboring word, and each entry their co-occurrence frequency. The raw counts are typically mapped into the components of a vector in some space using for example conditional probability, the log-likelihood ratio or tf-idf weighting. Under this representation, the similarity of word meanings can be easily quantified by measuring their distance in the vector space, the cosine of the angle between them, or their scalar product.

Our model assumes the same type of input data, namely a co-occurrence matrix, where rows correspond to *target* words and columns to *context features* (e.g., co-occurring neighbors). Throughout this paper we will use the notation t_i with $i : 1..I$ to refer to a target word and c_j with $j : 1..J$ to refer to context features. A cell (i, j) in the matrix represents the frequency of occurrence of target t_i with context feature c_j over a corpus.

Meaning Representation over Latent Senses We further assume that the target words t_i $i : 1...I$ found in a corpus share a global set of meanings or senses $Z = \{z_k | k : 1...K\}$. And therefore the meaning of individual target words can be described as a distribution over this set of senses. More formally, a target t_i is represented by the following vector:

$$\mathbf{v(t_i) = (P(z_1|t_i), ..., P(z_K|t_i))} \qquad (1)$$

where component $P(z_1|t_i)$ is the probability of sense z_1 given target word t_i, component $P(z_2|t_i)$ the probability of sense z_2 given t_i and so on.

The intuition behind such a representation is that a target word can be described by a set of core meanings and by the frequency with which these are attested. Note that the representation in (1) is not fixed but parametrized with respect to an input corpus (i.e., it only reflects word usage as attested in that corpus). The senses $z_1 \dots z_K$ are latent and can be seen as a means of reducing the dimensionality of the original co-occurrence matrix.

Analogously, we can represent the meaning of a target word given a context feature as:

$$\mathbf{v(t_i, c_j) = (P(z_1|t_i, c_j), ..., P(z_K|t_i, c_j))} \qquad (2)$$

Here, target t_i is again represented as a distribution over senses, but is now modulated by a specific context c_j which reflects actual word usage. This distribution is more "focused" compared to (1); the context helps disambiguate the meaning of the target word, and as a result fewer senses will share most of the probability mass.

In order to create the context-aware representations defined in (2) we must estimate the probabilities $P(z_k|t_i, c_j)$ which can be factorized as the product of $P(t_i, z_k)$, the joint probability of target t_i and latent sense z_k, and $P(c_j|z_k, t_i)$, the conditional probability of context c_j given target t_i and sense z_k:

$$P(z_k|t_i, c_j) = \frac{P(t_i, z_k)P(c_j|z_k, t_i)}{\sum_k P(t_i, z_k)P(c_j|z_k, t_i)} \qquad (3)$$

Problematically, the term $P(c_j|z_k, t_i)$ is difficult to estimate since it implies learning a total number of $K \times I$ J-dimensional distributions. We will therefore make the simplifying assumption that target words t_i and context features c_j are conditionally independent given sense z_k:

$$P(z_k|t_i, c_j) \approx \frac{P(z_k|t_i)P(c_j|z_k)}{\sum_k P(z_k|t_i)P(c_j|z_k)} \qquad (4)$$

Although not true in general, the assumption is relatively weak. We do not assume that words and context features occur independently of each other, but only that they are generated independently given an assigned meaning. A variety of latent variable models can be used to obtain senses $z_1 \dots z_K$ and estimate the distributions $P(z_k|t_i)$ and $P(c_j|z_k)$; we give specific examples in Section 4.

Note that we abuse terminology here, as the senses our models obtain are not lexicographic meaning distinctions. Rather, they denote coarse-grained senses or more generally topics attested in the document collections our model is trained on. Furthermore, the senses are not word-specific but global (i.e., shared across all words) and modulated either within or out of context probabilistically via estimating $P(z_k|t_i, c_j)$ and $P(z_k|t_i)$, respectively.

4 Parametrizations

The general framework outlined above can be parametrized with respect to the input co-occurrence matrix and the algorithm employed for inducing the latent structure. Considerable latitude is available when creating the co-occurrence matrix, especially when defining its columns, i.e., the linguistic contexts a target word is attested with. These contexts can be a small number of words surrounding the target word (Lund and Burgess, 1996; Lowe and McDonald, 2000), entire paragraphs, documents (Salton et al., 1975; Landauer and Dumais, 1997) or even syntactic dependencies (Grefenstette, 1994; Lin, 1998; Padó and Lapata, 2007).

Analogously, a number of probabilistic models can be employed to induce the latent senses. Examples include Probabilistic Latent Semantic Analysis (PLSA, Hofmann (2001)), Probabilistic Principal Components Analysis (Tipping and Bishop, 1999), non-negative matrix factorization (NMF, Lee and Seung (2000)), and latent Dirichlet allocation (LDA, Blei et al. (2003)). We give a more detailed description of the latter two models as we employ them in our experiments.

Non-negative Matrix Factorization Non-negative matrix factorization algorithms approximate a non-negative input matrix V by two non-negative factors W and H, under a given loss function. W and H are reduced-dimensional matrices and their product can be regarded as a compressed form of the data in V:

$$V_{I,J} \approx W_{I,K} H_{K,J} \tag{5}$$

where W is a basis vector matrix and H is an encoded matrix of the basis vectors in equation (5). Several loss functions are possible, such as mean squared error and Kullback-Leibler (KL) divergence. In keeping with the formulation in Section 3 we opt for a probabilistic interpretation of NMF (Gaussier and Goutte, 2005; Ding et al., 2008) and thus minimize the KL divergence between WH and V.

$$\min \sum_{i,j} (V_{i,j} \log \frac{V_{i,j}}{WH_{i,j}} - V_{i,j} + WH_{i,j}) \tag{6}$$

Specifically, we interpret matrix V as $V_{ij} = P(t_i, c_j)$, and matrices W and H as $P(t_i, z_k)$ and $P(c_j|z_k)$, respectively. We can also obtain the following more detailed factorization: $P(t_i, c_j) = \sum_k P(t_i)P(z_k|t_i)P(c_j|z_k)$.

Le WH denote the factors in a NMF decomposition of an input matrix V and B be a diagonal matrix with $B_{kk} = \sum_j H_{kj}$. $B^{-1}H$ gives a row-normalized version of H. Similarly, given matrix WB, we can define a diagonal matrix A, with $A_{ii} = \sum_k (WB)_{ik}$. $A^{-1}WB$ row-normalizes matrix WB. The factorization WH can now be rewritten as:

$$WH = AA^{-1}WBB^{-1}H = A(A^{-1}WB)(B^{-1}H)$$

which allows us to interpret A as $P(t_i)$, $A^{-1}WB$ as $P(z_k|t_i)$ and $B^{-1}H$ as $P(c_j|z_k)$. These interpretations are valid since the rows of $A^{-1}WB$ and of $B^{-1}H$ sum to 1, matrix A is diagonal with trace 1 because elements in WH sum to 1, and all entries are non-negative.

Latent Dirichlet Allocation LDA (Blei et al., 2003) is a probabilistic model of text generation. Each document d is modeled as a distribution over K topics, which are themselves characterized by distributions over words. The individual words in a document are generated by repeatedly sampling a topic according to the topic distribution and then sampling a single word from the chosen topic.

More formally, we first draw the mixing proportion over topics θ_d from a Dirichlet prior with parameters α. Next, for each of the N_d words w_{dn} in document d, a topic z_{dn} is first drawn from a multinomial distribution with parameters θ_{dn}. The probability of a word token w taking on value i given that topic $z = j$ is parametrized using a matrix β with $b_{ij} = P(w = i|z = j)$. Integrating out θ_d's and z_{dn}'s, gives $P(D|\alpha, \beta)$, the probability of a corpus (or document collection):

$$\prod_{d=1}^{M} \int P(\theta_d|\alpha) \left(\prod_{n=1}^{N_d} \sum_{z_{dn}} P(z_{dn}|\theta_d) P(w_{dn}|z_{dn}, \beta) \right) d\theta_d$$

The central computational problem in topic modeling is to obtain the posterior distribution $P(\theta, \mathbf{z}|\mathbf{w}, \alpha, \beta)$ of the hidden variables $\mathbf{z} = (z_1, z_2, \ldots, z_N)$. given a document $\mathbf{w} = (w_1, w_2, \ldots, w_N)$. Although this distribution is intractable in general, a variety of approximate inference algorithms have been proposed in the literature. We adopt the Gibbs sampling procedure discussed in Griffiths and Steyvers (2004). In this model, $P(w = i|z = j)$ is also a Dirichlet mixture (denoted ϕ) with symmetric priors (denoted β).

We use LDA to induce senses of target words based on context words, and therefore each row t_i in the input matrix transforms into a document. The frequency of t_i occurring with context feature c_j is the number of times word c_j is encountered in the "document" associated with t_i. We train the LDA model on this data to obtain the θ and ϕ distributions. θ gives the sense distributions of each target t_i: $\theta_{ik} = P(z_k|t_i)$ and ϕ the context-word distribution for each sense z_k: $\phi_{kj} = P(c_j|z_k)$.

5 Experimental Set-up

In this section we discuss the experiments we performed in order to evaluate our model. We describe the tasks on which it was applied, the corpora used for model training and our evaluation methodology.

Tasks The probabilistic model presented in Section 3 represents words via a set of induced senses. We experimented with two types of semantic space based on NMF and LDA and optimized parameters for these models on a word similarity task. The latter involves judging the similarity $sim(t_i, t'_i) = sim(v(t_i), v(t'_i))$ of words t_i and t'_i out of context, where $v(t_i)$ and $v(t'_i)$ are obtained from the output of NMF or LDA, respectively. In our experiments we used the data set of Finkelstein et al. (2002). It contains 353 pairs of words and their similarity scores as perceived by human subjects.

The contextualized representations were next evaluated on lexical substitution (McCarthy and Navigli, 2007). The task requires systems to find appropriate substitutes for target words occurring in context. Typically, systems are given a set of substitutes, and must produce a ranking such that appropriate substitutes are assigned a higher rank compared to non-appropriate ones. We made use of the SemEval 2007 Lexical Substitution Task benchmark data set. It contains 200 target words, namely nouns, verbs, adjectives and adverbs, each of which occurs in 10 distinct sentential contexts. The total set contains 2,000 sentences. Five human annotators were asked to provide substitutes for these target words. Table 1 gives an example of the adjective *still* and its substitutes.

Following Erk and Padó (2008), we pool together the total set of substitutes for each target word. Then, for each instance the model has to produce a ranking for the total substitute set. We rank the candidate substitutes based on the similarity of the contextualized target and the out-of-context substitute, $sim(v(t_i, c_j), v(t'_i))$, where t_i is the target word, c_j a context word and t'_i a substitute. Contextualizing just one of the words brings higher discriminative power to the model rather than performing comparisons with the target and its substitute embedded in an identical context (see also Thater et al. (2010) for a similar observation).

Sentences	Substitutes
It is important to apply the herbicide on a *still* day, because spray drift can kill non-target plants.	calm (5) not-windy (1) windless (1)
A movie is a visual document comprised of a series of *still* images.	motionless (3) unmoving (2) fixed (1) stationary (1) static (1)

Table 1: Lexical substitution data example for the adjective *still*; numbers in parentheses indicate the frequency of the substitute.

Model Training All the models we experimented with use identical input data, i.e., a bag-of-words matrix extracted from the GigaWord collection of news text. Rows in this matrix are target words and columns are their co-occurring neighbors, within a symmetric window of size 5. As context words, we used a vocabulary of the 3,000 most frequent words in the corpus.[1]

We implemented the classical NMF factorization algorithm described in Lee and Seung (2000). The input matrix was normalized so that all elements summed to 1. We experimented with four dimensions K: $[600 - 1000]$ with step size 200. We ran the algorithm for 150 iterations to obtain factors W and H which we further processes as described in Section 4 to obtain the desired probability distributions. Since the only parameter of the NMF model is the factorization dimension K, we performed two independent runs with each K value and averaged their predictions.

The parameters for the LDA model are the number of topics K and Dirichlet priors α and β. We experimented with topics K: $[600 - 1400]$, again with step size 200. We fixed β to 0.01 and tested two values for α: $\frac{2}{K}$ (Porteous et al., 2008) and $\frac{50}{K}$ (Griffiths and Steyvers, 2004). We used Gibbs sampling on the "document collection" obtained from the input matrix and estimated the sense distributions as described in Section 4. We ran the chains for 1000 iter-

[1] The GigaWord corpus contains 1.7B words; we scale down all the counts by a factor of 70 to speed up the computation of the LDA models. All models use this reduced size input data.

ations and averaged over five iterations $[600-1000]$ at lag 100 (we observed no topic drift).

We measured similarity using the scalar product, cosine, and inverse Jensen-Shannon (IJS) divergence (see (7), (8), and (9), respectively):

$$\mathrm{sp}(v,w) =< v,w >= \sum_i v_i w_i \qquad (7)$$

$$\cos(v,w) = \frac{\langle v,w\rangle}{||v||\,||w||} \qquad (8)$$

$$\mathrm{IJS}(v,w) = \frac{1}{\mathrm{JS(v,w)}} \qquad (9)$$

$$\mathrm{JS(v,w)} = \frac{1}{2}\mathrm{KL}(v|m) + \frac{1}{2}\mathrm{KL}(w|m) \qquad (10)$$

where m is a shorthand for $\frac{1}{2}(v+w)$ and KL the Kullback-Leibler divergence, $\mathrm{KL}(v|w) = \sum_i v_i log(\frac{v_i}{w_i})$.

Among the above similarity measures, the scalar product has the most straightforward interpretation as the probability of two targets sharing a common meaning (i.e., the sum over all possible meanings). The scalar product assigns 1 to a pair of identical vectors if and only if $P(z_i) = 1$ for some i and $P(z_j) = 0, \forall j \neq i$. Thus, only fully disambiguated words receive a score of 1. Beyond similarity, the measure also reflects how "focused" the distributions in question are, as very ambiguous words are unlikely to receive high scalar product values.

Given a set of context words, we contextualize the target using one context word at a time and compute the overall similarity score by multiplying the individual scores.

Baselines Our baseline models for measuring similarity out of context are Latent Semantic Analysis (Landauer and Dumais, 1997) and a simple semantic space without any dimensionality reduction.

For LSA, we computed the $U\Sigma V$ SVD decomposition of the original matrix to rank $k = 1000$. Any decomposition of lower rank can be obtained from this by setting rows and columns to 0. We evaluated decompositions to ranks K: [200 − 1000], at each 100 step. Similarity computations were performed in the lower rank approximation matrix $U\Sigma V$, as originally proposed in Deerwester et al. (1990), and in matrix U which maps the words into the concept space. It is common to compute SVD decompositions on matrices to which prior weighting schemes have been applied. We experimented with tf-idf weighting and line normalization.

Our second baseline, the simple semantic space, was based on the original input matrix on which we applied several weighting schemes such as point-wise mutual information, tf-idf, and line normalization. Again, we measured similarity using cosine, scalar product and inverse JS divergence. In addition, we also experimented with Lin's (1998) similarity measure:

$$\mathrm{lin}(v,w) = \frac{\sum_{i\in I(v)\cap I(w)}(v_i + w_i)}{\sum_{i\in I(v)} v_i + \sum_{l\in I(w)} w_i} \qquad (11)$$

where the values in v and w are point-wise mutual information, and $I(\cdot)$ gives the indices of positive values in a vector.

Our baselines for contextualized similarity were vector addition and vector multiplication which we performed using the simple semantic space (Mitchell and Lapata, 2008) and dimensionality reduced representations obtained from NMF and LDA. To create a ranking of the candidate substitutes we compose the vector of the target with its context and compare it with each substitute vector. Given a set of context words, we contextualize the target using each context word at a time and multiply the individual scores.

Evaluation Method For the word similarity task we used correlation analysis to examine the relationship between the human ratings and their corresponding vector-based similarity values. We report Spearman's ρ correlations between the similarity values provided by the models and the mean participant similarity ratings in the Finkelstein et al. (2002) data set. For the lexical substitution task, we compare the system ranking with the gold standard ranking using Kendall's τ_b rank correlation (which is adjusted for tied ranks). For all contextualized models we defined the context of a target word as the words occurring within a symmetric context window of size 5. We assess differences between models using stratified shuffling (Yeh, 2000).[2]

[2] Given two system outputs, the null hypothesis (i.e., that the two predictions are indistinguishable) is tested by randomly mixing the individual instances (in our case sentences) of the two outputs. We ran a standard number of 10000 iterations.

Model	Spearman ρ
SVS	38.35
LSA	49.43
NMF	**52.99**
LDA	**53.39**
LSA$_{\text{MIX}}$	49.76
NMF$_{\text{MIX}}$	51.62
LDA$_{\text{MIX}}$	51.97

Table 2: Results on out of context word similarity using a simple co-occurrence based vector space model (SVS), latent semantic analysis, non-negative matrix factorization and latent Dirichlet allocation as individual models with the best parameter setting (LSA, NMF, LDA) and as mixtures (LSA$_{\text{MIX}}$, NMF$_{\text{MIX}}$, LDA$_{\text{MIX}}$).

6 Results

Word Similarity Our results on word similarity are summarized in Table 2. The simple co-occurrence based vector space (SVS) performed best with tf-idf weighting and the cosine similarity measure. With regard to LSA, we obtained best results with initial line normalization of the matrix, $K = 600$ dimensions, and the scalar product similarity measure while performing computations in matrix U. Both NMF and LDA models are generally better with a larger number of senses. NMF yields best performance with $K = 1000$ dimensions and the scalar product similarity measure. The best LDA model also uses the scalar product, has $K = 1200$ topics, and α set to $\frac{50}{K}$.

Following Reisinger and Mooney (2010), we also evaluated mixture models that combine the output of models with varying parameter settings. For both NMF and LDA we averaged the similarity scores returned by all runs. For comparison, we also present an LSA mixture model over the (best) middle interval K values. As can be seen, the LSA model improves slightly, whereas NMF and LDA perform worse than their best individual models.[3] Overall, we observe that NMF and LDA yield significantly ($p < 0.01$) better correlations than LSA and the simple semantic space, both as individual models and as mixtures.

Model	Kendall's τ_b
SVS	11.05
Add-SVS	12.74
Add-NMF	12.85
Add-LDA	12.33
Mult-SVS	14.41
Mult-NMF	13.20
Mult-LDA	12.90
Cont-NMF	14.95
Cont-LDA	13.71
Cont-NMF$_{\text{MIX}}$	**16.01**
Cont-LDA$_{\text{MIX}}$	**15.53**

Table 3: Results on lexical substitution using a simple semantic space model (SVS), additive and multiplicative compositional models with vector representations based on co-occurrences (Add-SVS, Mult-SVS), NMF (Add-NMF, Mult-NMF), and LDA (Add-LDA, Mult-LDA) and contextualized models based on NMF and LDA with the best parameter setting (Cont-NMF, Cont-LDA) and as mixtures (Cont-NMF$_{\text{MIX}}$, Cont-LDA$_{\text{MIX}}$).

Lexical Substitution Our results on lexical substitution are shown in Table 3. As a baseline we also report the performance of the simple semantic space that does not use any contextual information. This model returns the same ranking of the substitute candidates for each instance, based solely on their similarity with the target word. This is a relatively competitive baseline as observed by Erk and Padó (2008) and Thater et al. (2009).

We report results with contextualized NMF and LDA as individual models (the best word similarity settings) and as mixtures (as described above). These are in turn compared against additive and multiplicative compositional models. We implemented an additive model with pmi weighting and Lin's similarity measure which is defined in an additive fashion. The multiplicative model uses tf-idf weighting and cosine similarity, which involves multiplication of vector components. Other combinations of weighting schemes and similarity measures delivered significantly lower results. We also report results for these models when using the NMF and LDA reduced representations.

[3] It is difficult to relate our results to Reisinger and Mooney (2010), due to differences in the training data and the vector representations it gives rise to. As a comparison, a baseline configuration with tf-idf weighting and the cosine similarity measure yields a correlation of 0.38 with our data and 0.49 in Reisinger and Mooney (2010).

Model	Adv	Adj	Noun	Verb
SVS	22.47	14.38	09.52	7.98
Add-SVS	22.79	14.56	11.59	10.00
Mult-SVS	22.85	16.37	13.59	11.60
Cont-NMF$_{\text{MIX}}$	**26.13**	**17.10**	15.16	**14.18**
Cont-LDA$_{\text{MIX}}$	21.21	16.00	**16.31**	13.67

Table 4: Results on lexical substitution for different parts of speech with a simple semantic space model (SVS), two compositional models (Add-SVS, Mult-SVS), and contextualized mixture models with NMF and LDA (Cont-NMF$_{\text{MIX}}$, Cont-LDA$_{\text{MIX}}$), using Kendall's τ_b correlation coefficient.

All models significantly ($p < 0.01$) outperform the context agnostic simple semantic space (see SVS in Table 3). Mixture NMF and LDA models are significantly better than all variants of compositional models ($p < 0.01$); the individual models are numerically better, however the difference is not statistically significant. We also find that the multiplicative model using a simple semantic space (Mult-SVS) is the best performing compositional model, thus corroborating the results of Mitchell and Lapata (2009). Interestingly, dimensionality compositional models. This indicates that the better results we obtain are due to the probabilistic formulation of our contextualized model as a whole rather than the use of NMF or LDA. Finally, we observe that the Cont-NMF model is slightly better than Cont-LDA, however the difference is not statistically significant.

To allow comparison with previous results reported on this data set, we also used the Generalized Average Precision (GAP, Kishida (2005)) as an evaluation measure. GAP takes into account the order of candidates ranked correctly by a hypothetical system, whereas average precision is only sensitive to their relative position. The best performing models are Cont-NMF$_{\text{MIX}}$ and Cont-LDA$_{\text{MIX}}$ obtaining a GAP of 42.7% and 42.9%, respectively. Erk and Padó (2010) report a GAP of 38.6% on this data set with their best model.

Table 4 shows how the models perform across different parts of speech. While verbs and nouns seem to be most difficult, we observe higher gains from the use of contextualized models. Cont-LDA$_{\text{MIX}}$ obtains approximately 7% absolute gain for nouns and Cont-NMF$_{\text{MIX}}$ approximately 6% for verbs. All contextualized models obtain smaller improvements for adjectives. For adverbs most models do not improve over the no-context setting, with the exception Cont-NMF$_{\text{MIX}}$.

Senses		Word Distributions
TRAFFIC	(0.18)	*road, traffic, highway, route, bridge*
MUSIC	(0.04)	*music, song, rock, band, dance, play*
FAN	(0.04)	*crowd, fan, people, wave, cheer, street*
VEHICLE	(0.04)	*car, truck, bus, train, driver, vehicle*

Table 5: Induced senses of *jam* and five most likely words given these senses using an LDA model; sense probabilities are shown in parentheses.

Finally, we also qualitatively examined how the context words influence the sense distributions of target words using examples from the lexical substitution dataset and the output of an individual Cont-LDA model. In many cases, a target word starts with a distribution spread over a larger number of senses, while a context word shifts this distribution to one majority sense. Consider, for instance, the target noun *jam* in the following sentence:

(1) With their transcendent, improvisational jams and Mayan-inspired sense of a higher, metaphysical purpose, the band's music delivers a spiritual sustenance that has earned them a very devoted core following.

Table 5 shows the out-of-context senses activated for *jam* together with the five most likely words associated with them.[4] Sense probabilities are also shown in parentheses. As can be seen, initially two traffic-related and two music-related senses are activated, however with low probabilities. In the presence of the context word *band*, we obtain a much more "focused" distribution, in which the MUSIC sense has 0.88 probability. The system ranks *riff* and *gig* as the most likely two substitutes for *jam*. The gold annotation also lists *session* as a possible substitute.

In a large number of cases, the target is only partially disambiguated by a context word and this is also reflected in the resulting distribution. An ex-

[4] Sense names are provided by the authors in an attempt to best describe the clusters (i.e., topics for LDA) to which words are assigned.

ample is the word *bug* which initially has a distribution triggering the SOFTWARE (0.09, *computer, software, microsoft, windows*) and DISEASE (0.06, *disease, aids, virus, cause*) senses. In the context of *client*, *bug* remains ambiguous between the senses SECRET-AGENCY (0.34, *agent, secret, intelligence, FBI)*) and SOFTWARE (0.29):

(2) We wanted to give our client more than just a list of bugs and an invoice — we wanted to provide an audit trail of our work along with meaningful productivity metrics.

There are also cases where the contextualized distributions are not correct, especially when senses are domain specific. An example is the word *function* occurring in its mathematical sense with the context word *distribution*. However, the senses that are triggered by this pair all relate to the "service" sense of *function*. This is a consequence of the newspaper corpus we use, in which the mathematical sense of *function* is rare. We also see several cases where the target word and one of the context words are assigned senses that are locally correct, but invalid in the larger context. In the following example:

(3) Check the shoulders so it hangs well, stops at hips or below, and make sure the pants are long enough.

The pair *(check, shoulder)* triggers senses INJURY (0.81, *injury, left, knee, shoulder*) and BALL-SPORTS (0.10, *ball, shot, hit, throw*). However, the sentential context ascribes a meaning that is neither related to injury nor sports. This suggests that our models could benefit from more principled context feature aggregation.

Generally, verbs are not as good context words as nouns. To give an example, we often encounter the pair *(let, know)*, used in the common "inform" meaning. The senses we obtain for this pair, are, however, rather uninformative general verb classes: {*see, know, think, do*} (0.57) and {*go, say, do, can*} (0.20). This type of error can be eliminated in a space where context features are designed to best reflect the properties of the target words.

7 Conclusions

In this paper we have presented a general framework for computing similarity in context. Key in this framework is the representation of word meaning as a distribution over a set of global senses where contextualized meaning is modeled as a change in this distribution. The approach is conceptually simple, the same vector representation is used for isolated words and words in context without being tied to a specific sense induction method or type of semantic space.

We have illustrated two instantiations of this framework using non-negative matrix factorization and latent Dirichlet allocation for inducing the latent structure, and shown experimentally that they outperform previously proposed methods for measuring similarity in context. Furthermore, both of them benefit from mixing model predictions over a set of different parameter choices, thus making parameter tuning redundant.

The directions for future work are many and varied. Conceptually, we have defined our model in an asymmetric fashion, i.e., by stipulating a difference between target words and contextual features. However, in practice, we used vector representations that do not distinguish the two: target words and contextual features are both words. This choice was made to facilitate comparisons with the popular bag-of-words vector space models. However, differentiating target from context representations may be beneficial particularly when the similarity computations are embedded within specific tasks such as the acquisition of paraphrases, the recognition of entailment relations, and thesaurus construction. Also note that our model currently contextualizes target words with respect to individual contexts. Ideally, we would like to compute the collective influence of several context words on the target. We plan to further investigate how to select or to better aggregate the entire set of features extracted from a context.

Acknowledgments The authors acknowledge the support of the DFG (Dinu; International Research Training Group "Language Technology and Cognitive Systems") and EPSRC (Lapata; grant GR/T04540/01).

References

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Daoud Clarke. 2009. Context-theoretic semantics for natural language: an overview. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 112–119, Athens, Greece.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407.

Chris Ding, Tao Li, and Wei Peng. 2008. On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Computational Statistics & Data Analysis*, 52(8):3913–3927.

Katrin Erk and Sabastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 897–906, Honolulu, Hawaii.

Katrin Erk and Sebastian Padó. 2010. Exemplar-based models for word meaning in context. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 92–97, Uppsala, Sweden.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: the concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

Eric Gaussier and Cyril Goutte. 2005. Relation between PLSA and NMF and implications. In *Proceedings of the 28th Annual international ACM SIGIR conference on Research and development in information retrieval*, pages 601–602, New York, NY.

Gregory Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl. 1):5228–5235.

Thomas Hofmann. 2001. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 41(2):177–196.

Walter Kintsch. 2001. Predication. *Cognitive Science*, 25:173–202.

Kazuaki Kishida. 2005. Property of average precision and its generalization: An examination of evaluation indicator for information retrieval experiments. *NII Technical Report*.

Thomas K. Landauer and Susan T. Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104(2):211–240.

Daniel D. Lee and H. Sebastian Seung. 2000. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):342–360.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the joint Annual Meeting of the Association for Computational Linguistics and International Conference on Computational Linguistics*, pages 768–774, Montréal, Canada.

Will Lowe and Scott McDonald. 2000. The direct route: Mediated priming in semantic space. In *Proceedings of the 22nd Annual Conference of the Cognitive Science Society*, pages 675–680, Philadelphia, PA.

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, and Computers*, 28:203–208.

Diana McCarthy and Roberto Navigli. 2007. SemEval-2007 Task 10: English Lexical Substitution Task. In *Proceedings of SemEval*, pages 48–53, Prague, Czech Republic.

Scott McDonald. 2000. *Environmental Determinants of Lexical Processing Effort*. Ph.D. thesis, University of Edinburgh.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244, Columbus, Ohio.

Jeff Mitchell and Mirella Lapata. 2009. Language models based on semantic composition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 430–439, Suntec, Singapore.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. 2008. Fast collapsed gibbs sampling for latent Dirichlet allocation. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 569–577, New York, NY.

Joseph Reisinger and Raymond J. Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117, Los Angeles, California.

G Salton, A Wang, and C Yang. 1975. A vector-space model for information retrieval. *Journal of the American Society for Information Science*, 18:613–620.

Hinrich Schuetze. 1998. Automatic word sense discrimination. *Journal of Computational Linguistics*, 24:97–123.

Stefan Thater, Georgiana Dinu, and Manfred Pinkal. 2009. Ranking paraphrases in context. In *Proceedings of the 2009 Workshop on Applied Textual Inference*, pages 44–47, Suntec, Singapore.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2010. Contextualizing semantic representations using syntactically enriched vector models. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 948–957, Uppsala, Sweden.

Michael E. Tipping and Chris M. Bishop. 1999. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 61:611–622.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th Conference on Computational Linguistics*, pages 947–953, Saarbrücken, Germany.

A Mixture Model with Sharing for Lexical Semantics

Joseph Reisinger
Department of Computer Science
University of Texas at Austin
1616 Guadalupe, Suite 2.408
Austin, TX, 78701
joeraii@cs.utexas.edu

Raymond Mooney
Department of Computer Science
University of Texas at Austin
1616 Guadalupe, Suite 2.408
Austin, TX, 78701
mooney@cs.utexas.edu

Abstract

We introduce *tiered clustering*, a mixture model capable of accounting for varying degrees of shared (context-independent) feature structure, and demonstrate its applicability to inferring distributed representations of word meaning. Common tasks in lexical semantics such as word relatedness or selectional preference can benefit from modeling such structure: Polysemous word usage is often governed by some common background metaphoric usage (e.g. the senses of *line* or *run*), and likewise modeling the selectional preference of verbs relies on identifying commonalities shared by their typical arguments. Tiered clustering can also be viewed as a form of soft feature selection, where features that do not contribute meaningfully to the clustering can be excluded. We demonstrate the applicability of tiered clustering, highlighting particular cases where modeling shared structure is beneficial and where it can be detrimental.

1 Introduction

Word meaning can be represented as high-dimensional vectors inhabiting a common space whose dimensions capture semantic or syntactic properties of interest (e.g. Erk and Pado, 2008; Lowe, 2001). Such *vector-space* representations of meaning induce measures of word similarity that can be tuned to correlate well with judgements made by humans. Previous work has focused on designing feature representations and semantic spaces that capture salient properties of word meaning (e.g. Curran, 2004; Gabrilovich and Markovitch, 2007; Landauer and Dumais, 1997), often leveraging the distributional hypothesis, i.e. that similar words appear in similar contexts (Miller and Charles, 1991; Pereira et al., 1993).

Since vector-space representations are constructed at the lexical level, they conflate multiple word meanings into the same vector, e.g. collapsing occurrences of *bank*$_{\text{institution}}$ and *bank*$_{\text{river}}$. Methods such as *Clustering by Committee* (Pantel, 2003) and *multi-prototype* representations (Reisinger and Mooney, 2010) address this issue by performing word-sense disambiguation across word occurrences, and then building meaning vectors from the disambiguated words. Such approaches can readily capture the structure of homonymous words with several unrelated meanings (e.g. *bat* and *club*), but are not suitable for representing the common metaphor structure found in highly polysemous words such as *line* or *run*.

In this paper, we introduce *tiered clustering*, a novel probabilistic model of the *shared structure* often neglected in clustering problems. Tiered clustering performs *soft* feature selection, allocating features between a Dirichlet Process clustering model and a background model consisting of a single component. The background model accounts for features commonly shared by all occurrences (i.e. context-independent feature variation), while the clustering model accounts for variation in word usage (i.e. context-dependent variation, or *word senses*; Table 1).

Using the tiered clustering model, we derive a multi-prototype representation capable of capturing varying degrees of sharing between word senses, and demonstrate its effectiveness in lexical semantic tasks where such sharing is desirable. In particular we show that tiered clustering outperforms the multi-prototype approach for (1) selectional preference (Resnik, 1997; Pantel et al., 2007), i.e. predict-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1173–1182,
MIT, Massachusetts, USA, 9-11 October 2010.

ing the typical filler of an argument slot of a verb, and (2) word-relatedness in the presence of highly polysemous words. The former case exhibits a high degree of explicit structure, especially for more selectionally restrictive verbs (e.g. the set of things that can be *eaten* or can *shoot*).

The remainder of the paper is organized as follows: Section 2 gives relevant background on the methods compared, Section 3 outlines the multi-prototype model based on the Dirichlet Process mixture model, Section 4 derives the tiered clustering model, Section 5 discusses similarity metrics, Section 6 details the experimental setup and includes a micro-analysis of feature selection, Section 7 presents results applying tiered clustering to word relatedness and selectional preference, Section 8 discusses future work, and Section 9 concludes.

2 Background

Models of the *attributional similarity* of concepts, i.e. the degree to which concepts overlap based on their attributes (Turney, 2006), are commonly implemented using vector-spaces derived from (1) word collocations (Schütze, 1998), directly leveraging the distributional hypothesis (Miller and Charles, 1991), (2) syntactic relations (Padó and Lapata, 2007), (3) structured corpora (e.g. Gabrilovich and Markovitch (2007)) or (4) latent semantic spaces (Finkelstein et al., 2001; Landauer and Dumais, 1997). Such models can be evaluated based on their correlation with human-reported lexical similarity judgements using e.g. the WordSim-353 collection (Finkelstein et al., 2001). Distributional methods exhibit a high degree of scalability (Gorman and Curran, 2006) and have been applied broadly in information retrieval (Manning et al., 2008), large-scale taxonomy induction (Snow et al., 2006), and knowledge acquisition (Van Durme and Paşca, 2008).

Reisinger and Mooney (2010) introduced a *multi-prototype* approach to vector-space lexical semantics where individual words are represented as collections of "prototype" vectors. This representation is capable of accounting for homonymy and polysemy, as well as other forms of variation in word usage, like similar context-dependent methods (Erk and Pado, 2008). The set of vectors for a word is determined by unsupervised *word sense discovery* (Schütze, 1998), which clusters the contexts in which a word appears. Average prototype vectors are then computed separately for each cluster, producing a distributed representation for each word.

LIFE	
	all, about, life, would, death
	my, you, real, your, about spent, years, rest, lived, last sentenced, imprisonment, sentence, prison insurance, peer, Baron, member, company Guru, Rabbi, Baba, la, teachings
RADIO	
	station, radio, stations, television
	amateur, frequency, waves, system show, host, personality, American song, single, released, airplay operator, contact, communications, message
WIZARD	
	evil, powerful, magic, wizard
	Merlin, King, Arthur, Arthurian fairy, wicked, scene, tale Harry, Potter, Voldemort, Dumbledore
STOCK	
	stock, all, other, company, new
	market, crash, markets, price, prices housing, breeding, fish, water, horses car, racing, cars, NASCAR, race, engine card, cards, player, pile, game, paper rolling, locomotives, line, new, railway

Table 1: Example tiered clustering representation of words with varying degrees of polysemy. Each boxed set shows the most common background (shared) features, and each prototype captures one thematic usage of the word. For example, *wizard* is broken up into a background cluster describing features common to all usages of the word (e.g., *magic* and *evil*) and several genre-specific usages (e.g. *Merlin*, *fairy tales* and *Harry Potter*).

Distributional methods have also proven to be a powerful approach to modeling *selectional preference* (Padó et al., 2007; Pantel et al., 2007), rivaling methods based on existing semantic resources such as WordNet (Clark and Weir, 2002; Resnik, 1997) and FrameNet (Padó, 2007) and performing nearly as well as supervised methods (Herdağdelen and Baroni, 2009). Selectional preference has been shown to be useful for, e.g., resolving ambiguous attachments (Hindle and Rooth, 1991), word sense disambiguation (McCarthy and Carroll, 2003) and semantic role labeling (Gildea and Jurafsky, 2002).

3 Multi-Prototype Models

Representing words as mixtures over several prototypes has proven to be a powerful approach to vector-space lexical semantics (Pantel, 2003; Pantel et al., 2007; Reisinger and Mooney, 2010). In this section we briefly introduce a version of the multi-prototype model based on the Dirichlet Process Mixture Model (DPMM), capable of inferring automatically the number of prototypes necessary for each word (Rasmussen, 2000). Similarity between two DPMM word-representations is then computed as a function of their cluster centroids (§5), instead of the centroid of all the word's occurrences.

Multiple prototypes for each word w are generated by clustering feature vectors $v(c)$ derived from each occurrence $c \in \mathcal{C}(w)$ in a large textual corpus and collecting the resulting cluster centroids $\pi_k(w), k \in [1, K_w]$. This approach is commonly employed in unsupervised word sense discovery; however, we do not assume that clusters correspond to word senses. Rather, we only rely on clusters to capture meaningful variation in word usage.

Instead of assuming all words can be represented by the same number of clusters, we allocate representational flexibility dynamically using the DPMM. The DPMM is an infinite capacity model capable of assigning data to a variable, but finite number of clusters K_w, with probability of assignment to cluster k proportional to the number of data points previously assigned to k. A single parameter η controls the degree of smoothing, producing more uniform clusterings as $\eta \to \infty$. Using this model, the number of clusters no longer needs to be fixed a priori, allowing the model to allocate expressivity dynamically to concepts with richer structure. Such a model naturally allows the word representation to allocate additional capacity for highly polysemous words, with the number of clusters growing logarithmically with the number of occurrences. The DPMM has been used for rational models of concept organization (Sanborn et al., 2006), but to our knowledge has not yet been applied directly to lexical semantics.

4 Tiered Clustering

Tiered clustering allocates features between two submodels: a (context-dependent) DPMM and a single (context-independent) *background* component. This model is similar structurally to the feature selective clustering model proposed by Law et al. (2002). However, instead of allocating entire feature *dimensions* between model and background components, assignment is done at the level of individual feature occurrences, much like topic assignment in Latent Dirichlet Allocation (LDA; Griffiths et al., 2007). At a high level, the tiered model can be viewed as a combination of a multi-prototype model and a single-prototype back-off model. However, by leveraging both representations in a joint framework, uninformative features can be removed from the clustering, resulting in more semantically tight clusters.

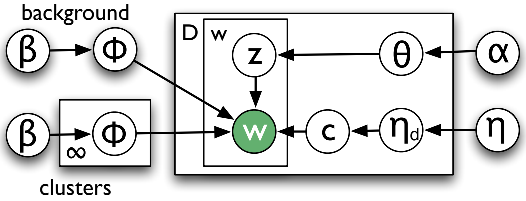


Figure 1: Plate diagram for the tiered clustering model with cluster indicators drawn from the Chinese Restaurant Process.

Concretely, each word occurrence $\mathbf{w}_d$ first selects a cluster $\boldsymbol{\phi}_d$ from the DPMM; then each feature $w_{i,d}$ is generated from either the background model $\boldsymbol{\phi}_{\text{back}}$ or the selected cluster $\boldsymbol{\phi}_d$, determined by the tier indicator $z_{i,d}$. The full generative model for tiered clustering is given by

$$\begin{array}{rcll}
\theta_d|\alpha & \sim & \text{Beta}(\alpha) & d \in D, \\
\boldsymbol{\phi}_d|\boldsymbol{\beta}, G_0 & \sim & \text{DP}(\boldsymbol{\beta}, G_0) & d \in D, \\
\boldsymbol{\phi}_{\text{back}}|\boldsymbol{\beta}_{\text{back}} & \sim & \text{Dirichlet}(\boldsymbol{\beta}_{\text{back}}) & \\
z_{i,d}|\theta_d & \sim & \text{Bernoulli}(\theta_d) & i \in |\mathbf{w}_d|, \\
w_{i,d}|\boldsymbol{\phi}_d, z_{i,d} & \sim & \begin{cases} \text{Mult}(\boldsymbol{\phi}_{\text{back}}) \\ \quad (z_{i,d} = 1) \\ \text{Mult}(\boldsymbol{\phi}_d) \\ \quad \text{(otherwise)} \end{cases} & i \in |\mathbf{w}_d|,
\end{array}$$

where $\boldsymbol{\alpha}$ controls the per-data tier distribution smoothing and $\boldsymbol{\beta}$ controls the uniformity of the DP cluster allocation. The DP is parameterized by a base measure G_0, controlling the per-cluster term distribution smoothing; which use a Dirichlet with hyperparameter η, as is common (Figure 1).

Since the background topic is shared across all occurrences, it can account for features with *context-independent* variance, such as stop words and other high-frequency noise, as well as the central tendency of the collection (Table 1). Furthermore, it is possible to put an asymmetric prior on η, yielding more fine-grained control over the assumed *uniformity* of the occurrence of noisy features, unlike in the model proposed by Law et al. (2002).

Although exact posterior inference is intractable in this model, we derive an efficient *collapsed Gibbs sampler* via analogy to LDA (Appendix 1).

5 Measuring Semantic Similarity

Due to its richer representational structure, computing similarity in the multi-prototype model is less straightforward than in the single prototype case. Reisinger and Mooney (2010) found that simply averaging all similarity scores over all pairs of prototypes (sampled from the cluster distributions) performs reasonably well and is robust to noise. Given two words w and w', this *AvgSim* metric is

$$\text{AvgSim}(w, w') \overset{\text{def}}{=} \frac{1}{K_w K_{w'}} \sum_{j=1}^{K_w} \sum_{k=1}^{K_{w'}} d(\pi_k(w), \pi_j(w'))$$

K_w and $K_{w'}$ are the number of clusters for w and w' respectively, and $d(\cdot, \cdot)$ is a standard distributional similarity measure (e.g. cosine distance). As cluster sizes become more uniform, AvgSim tends towards the single prototype similarity,[1] hence the effectiveness of AvgSim stems from boosting the influence of small clusters.

Tiered clustering representations offer more possibilities for computing semantic similarity than multi-prototype, as the background prototype can be treated separately from the other prototypes. We make use of a simple sum of the distance between the two background components, and the AvgSim of the two sets of clustering components.

6 Experimental Setup

6.1 Corpus

Word occurrence statistics are collected from a snapshot of English Wikipedia taken on Sept. 29th, 2009. Wikitext markup is removed, as are articles with fewer than 100 words, leaving 2.8M articles with a total of 2.05B words. Wikipedia was chosen due to its semantic breadth.

6.2 Evaluation Methodology

We evaluate the tiered clustering model on two problems from lexical semantics: word relatedness and selectional preference. For the word relatedness evaluation, we compared the predicted similarity of word pairs from each model to two collections of human similarity judgements: WordSim-353 (Finkelstein et al., 2001) and the Princeton Evocation relations (WN-Evocation, Ma et al., 2009).

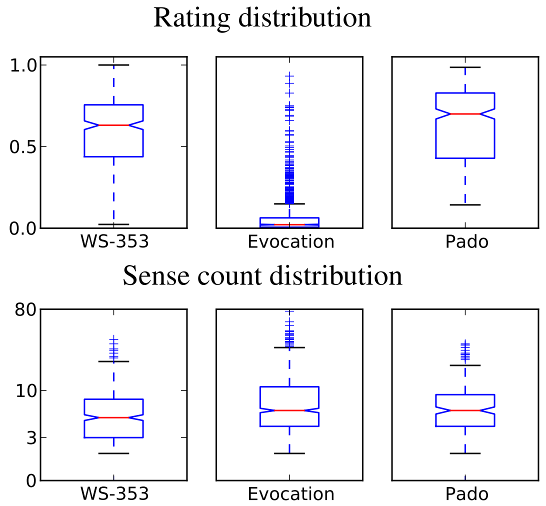


Figure 2: (**top**) The distribution of ratings (scaled [0,1]) on WS-353, WN-Evocation and Padó datasets. (**bottom**) The distribution of sense counts for each data set (log-domain), collected from WordNet 3.0.

WS-353 contains between 13 and 16 human similarity judgements for each of 353 word pairs, rated on a 1–10 integer scale. WN-Evocation is significantly larger than WS-353, containing over 100k similarity comparisons collected from trained human raters. Comparisons are assigned to only 3-5 human raters on average and contain a significantly higher fraction of zero- and low-similarity items than WS-353 (Figure 2), reflecting more accurately real-world lexical semantics settings. In our experiments we discard all comparisons with fewer than 5 ratings and then sample 10% of the remaining pairs uniformly at random, resulting in a test set with 1317 comparisons.

For selectional preference, we employ the *Padó* dataset, which contains 211 verb-noun pairs with human similarity judgements for how plausible the noun is for each argument of the verb (2 arguments per verb, corresponding roughly to subject and object). Results are averaged across 20 raters; typical inter-rater agreement is $\rho = 0.7$ (Padó et al., 2007).

In all cases correlation with human judgements is computed using Spearman's nonparametric rank correlation (ρ) with average human judgements

[1] This can be problematic for certain clustering methods that specify uniform priors over cluster sizes; however the DPMM naturally exhibits a linear decay in cluster sizes with the $\mathbb{E}[\#\text{ clusters of size } M] = \eta/M$.

(Agirre et al., 2009).

6.3 Feature Representation

In the following analyses we confine ourselves to representing word occurrences using unordered unigrams collected from a window of size T=10 centered around the occurrence, represented using *tf-idf* weighting. Feature vectors are pruned to a fixed length f, discarding all but the highest-weight features (f is selected via empirical validation, as described in the next section). Finally, semantic similarity between word pairs is computed using cosine distance (ℓ_2-normalized dot-product).[2]

6.4 Feature Pruning

Feature pruning is one of the most significant factors in obtaining high correlation with human similarity judgements using vector-space models, and has been suggested as one way to improve sense disambiguation for polysemous verbs (Xue et al., 2006). In this section, we calibrate the single prototype and multi-prototype methods on WS-353, reaching the limit of human and oracle performance and demonstrating robust performance gains even with semantically impoverished features. In particular we obtain ρ=0.75 correlation on WS-353 using *only* unigram collocations and ρ=0.77 using a fixed-K multi-prototype representation (Figure 3; Reisinger and Mooney, 2010). This result rivals average human performance, obtaining correlation near that of the supervised oracle approach of Agirre et al. (2009).

The optimal pruning cutoff depends on the feature weighting and number of prototypes as well as the feature representation. *t-test* and χ^2 features are most robust to feature noise and perform well even with no pruning; *tf-idf* yields the best results but is most sensitive to the pruning parameter (Figure 3). As the number of features increases, more pruning is required to combat feature noise.

Figure 4 breaks down the similarity pairs into four quantiles for each data set and then shows correlation separately for each quantile. In general the more polarized data quantiles (1 and 4) have higher correlation, indicating that fine-grained distinctions in semantic distance are easier for those sets.[3] Feature pruning improves correlations in quantiles 2–4 while reducing correlation in quantile 1 (lowest similarity). This result is to be expected as more features are necessary to make fine-grained distinctions between dissimilar pairs.

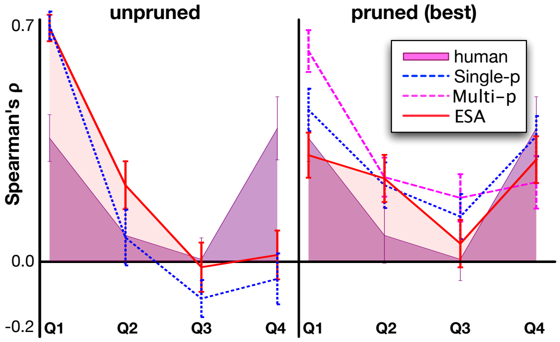


Figure 4: Correlation results on WS-353 broken down over quantiles in the human ratings. Quantile ranges are shown in Figure 2. In general ratings for highly similar (dissimilar) pairs are more predictable (quantiles 1 and 4) than middle similarity pairs (quantiles 2, 3). ESA shows results for a more semantically rich feature set derived using Explicit Semantic Analysis (Gabrilovich and Markovitch, 2007).

7 Results

We evaluate four models: (1) the standard single-prototype approach, (2) the DPMM multi-prototype approach outlined in §3, (3) a simple combination of the multi-prototype and single-prototype approaches (MP+SP)[4] and (4) the tiered clustering approach (§4). Each data set is divided into 5 quantiles based on per-pair average sense counts,[5] collected from WordNet 3.0 (Fellbaum, 1998); examples of pairs in the *high-polysemy* quantile are shown in Table 2. Unless otherwise specified, both DPMM multi-prototype and tiered clustering

[2] (**Parameter robustness**) We observe lower correlations on average for T=25 and T=5 and therefore observe T=10 to be near-optimal. Substituting weighted Jaccard similarity for cosine does not significantly affect the results in this paper.

[3] The fact that the per-quantile correlation is significantly lower than the full correlation e.g. in the human case indicates that fine-grained ordering (within quantile) is more difficult than coarse-grained (between quantile).

[4] (**MP+SP**) Tiered clustering's ability to model both shared and idiosyncratic structure can be easily approximated by using the single prototype model as the shared component and multi-prototype model as the clustering. However, unlike in the tiered model, all features are assigned to *both* components. We demonstrate that this simplification actually hurts performance.

[5] Despite many skewed pairs (e.g. line has 36 senses while insurance has 3), we found that arithmetic average and geometric average perform the same.

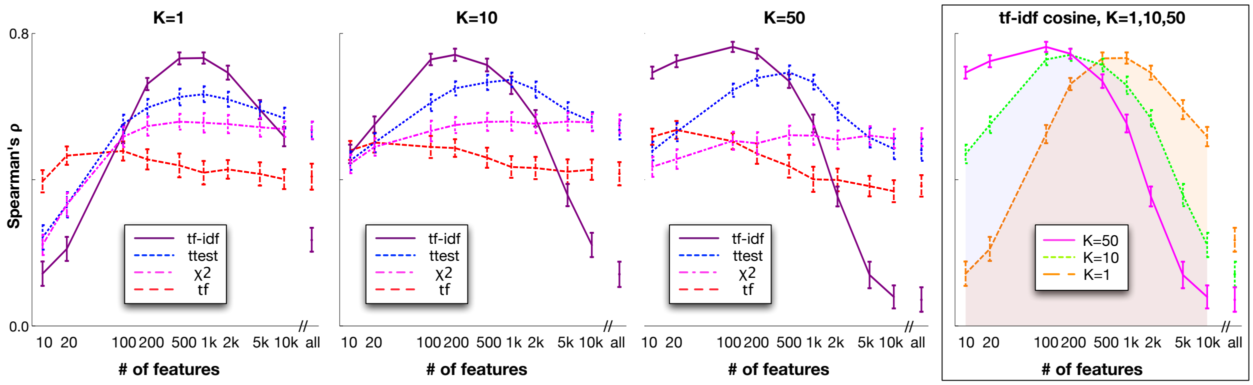


Figure 3: Effects of feature pruning and representation on WS-353 correlation broken down across multi-prototype representation size. In general *tf-idf* features are the most sensitive to pruning level, yielding the highest correlation for moderate levels of pruning and significantly lower correlation than other representations without pruning. The optimal amount of pruning varies with the number of prototypes used, with fewer features being optimal for more clusters. Bars show 95% confidence intervals.

WordSim-353
stock-live, start-match, line-insurance, game-round, street-place, company-stock
Evocation
break-fire, clear-pass, take-call, break-tin, charge-charge, run-heat, social-play
Padó
see-drop, see-return, hit-stock, raise-bank, see-face, raise-firm, raise-question

Table 2: Examples of highly polysemous pairs from each data set using sense counts from WordNet.

Method	$\rho \cdot 100$	$\mathbb{E}[C]$	background
Single prototype	73.4±0.5	1.0	-
high polysemy	76.0±0.9	1.0	-
Multi-prototype	76.8±0.4	14.8	-
high polysemy	79.3±1.3	12.5	-
MP+SP	75.4±0.5	14.8	-
high polysemy	80.1±1.0	12.5	-
Tiered	76.9±0.5	27.2	43.0%
high polysemy	83.1±1.0	24.2	43.0%

Table 3: Spearman's correlation on the WS-353 data set. *All* refers to the full set of pairs, *high polysemy* refers to the top 20% of pairs, ranked by sense count. $\mathbb{E}[C]$ is the average number of clusters employed by each method and *background* is the average percentage of features allocated by the tiered model to the background cluster. 95% confidence intervals are computed via bootstrapping.

use symmetric Dirichlet hyperparameters, β=0.1, η=0.1, and tiered clustering uses α=10 for the background/clustering allocation smoother.

7.1 WordSim-353

Correlation results for WS-353 are shown in Table 3. In general the approaches incorporating multiple prototypes outperform single prototype ($\rho = 0.768$ vs. $\rho = 0.734$). The tiered clustering model does not significantly outperform either the multi-prototype or MP+SP models on the full set, but yields significantly higher correlation on the high-polysemy set.

The tiered model generates more clusters than DPMM multi-prototype (27.2 vs. 14.8), despite using the same hyperparameter settings: Since words commonly shared across clusters have been allocated to the background component, the cluster components have less overlap and hence the model naturally allocates more clusters.

Examples of the tiered clusterings for several words from WS-353 are shown in Table 1 and corresponding clusters from the multi-prototype approach are shown in Table 4. In general the background component does indeed capture commonalities between all the sense clusters (e.g. all wizards use magic) and hence the tiered clusters are more semantically pure. This effect is most visible in *thematically polysemous* words, e.g. *radio* and *wizard*.

7.2 Evocation

Compared to WS-353, the WN-Evocation pair set is sampled more uniformly from English word pairs and hence contains a significantly larger fraction of unrelated words, reflecting the fact that word sim-

Word	Clusters
LIFE	my, you, real, about, your, would
	years, spent, rest, lived, last
	sentenced, imprisonment, sentence, prison
	years, cycle, life, all, expectancy, other
	all, life, way, people, human, social, many
RADIO	station, FM, broadcasting, format, AM
	radio, station, stations, amateur,
	show, station, host, program, radio
	stations, song, single, released, airplay
	station, operator, radio, equipment, contact
WIZARD	evil, magic, powerful, named, world
	Merlin, King, Arthur, powerful, court
	spells, magic, cast, wizard, spell, witch
	Harry, Dresden, series, Potter, character
STOCK	market, price, stock, company, value, crash
	housing, breeding, all, large, stock, many
	car, racing, company, cars, summer, NASCAR
	stock, extended, folded, card, barrel, cards
	rolling, locomotives, new, character, line

Table 4: Example DPMM multi-prototype representation of words with varying degrees of polysemy. Compared to the tiered clustering results in Table 1 the multi-prototype clusters are significantly less pure for *thematically polysemous* words such as *radio* and *wizard*.

Method	$\rho \cdot 100$	$\mathbb{E}[C]$	background
Single prototype	19.8±0.6	1.0	-
high similarity	23.9±1.1	1.0	-
high polysemy	11.5±1.2	1.0	-
Multi-prototype	20.1±0.5	14.8	-
high similarity	22.7±1.2	14.1	-
high polysemy	13.0±1.3	13.2	-
MP+SP	17.6±0.5	14.8	-
high similarity	23.5±1.2	14.1	-
high polysemy	11.4±1.0	13.2	-
Tiered	22.4±0.6	29.7	46.6%
high similarity	27.7±1.3	29.9	47.2%
high polysemy	15.4±1.1	27.4	46.6%

Table 5: Spearman's correlation on the Evocation data set. The *high similarity* subset contains the top 20% of pairs sorted by average rater score.

Method	$\rho \cdot 100$	$\mathbb{E}[C]$	background
Single prototype	25.8±0.8	1.0	-
high polysemy	17.3±1.7	1.0	-
Multi-prototype	20.2±1.0	18.5	-
high polysemy	14.1±2.4	17.4	-
MP+SP	19.7±1.0	18.5	-
high polysemy	10.5±2.5	17.4	-
Tiered	29.4±1.0	37.9	41.7%
high polysemy	28.5±2.4	37.4	43.2%

Table 6: Spearman's correlation on the Padó data set.

ilarity is a sparse relation (Figure 2 top). Furthermore, it contains proportionally more highly polysemous words relative to WS-353 (Figure 2 bottom).

On WN-Evocation, the single prototype and multi-prototype do not differ significantly in terms of correlation (ρ=0.198 and ρ=0.201 respectively; Table 5), while SP+MP yields significantly lower correlation (ρ=0.176), and the tiered model yields significantly higher correlation (ρ=0.224). Restricting to the top 20% of pairs with highest human similarity judgements yields similar outcomes, with single prototype, multi-prototype and SP+MP statistically indistinguishable (ρ=0.239, ρ=0.227 and ρ=0.235), and tiered clustering yielding significantly higher correlation (ρ=0.277). Likewise tiered clustering achieves the most significant gains on the high polysemy subset.

7.3 Selectional Preference

Tiered clustering is a natural model for verb selectional preference, especially for more selectionally restrictive verbs: the set of words that appear in a particular argument slot naturally have some kind of commonality (i.e. they can be *eaten* or can *promise*). The background component of the tiered clustering model can capture such general argument structure. We model each verb argument slot in the Padó set with a separate tiered clustering model, separating terms co-occurring with the target verb according to which slot they fill.

On the Padó set, the performance of the DPMM multi-prototype approach breaks down and it yields significantly lower correlation with human norms than the single prototype (ρ=0.202 vs. ρ=0.258; Table 6), due to its inability to capture the shared structure among verb arguments. Furthermore combining with the single prototype does not significantly change its performance (ρ=0.197). Moving to the tiered model, however, yields significant improvements in correlation over the other models (ρ=0.294), primarily improving correlation in the case of highly polysemous verbs and arguments.

8 Discussion and Future Work

We have demonstrated a novel model for distributional lexical semantics capable of capturing both shared (context-independent) and idiosyncratic (context-dependent) structure in a set of word occurrences. The benefits of this tiered model were most pronounced on a selectional preference task, where there is significant shared structure imposed by conditioning on the verb. Although our results on the Padó are not state of the art,[6] we believe this to be due to the impoverished vector-space design; tiered clustering can be applied to more expressive vector spaces, such as those incorporating dependency parse and FrameNet features.

One potential explanation for the superior performance of the tiered model vs. the DPMM multi-prototype model is simply that it allocates more clusters to represent each word (Reisinger and Mooney, 2010). However, we find that decreasing the hyperparameter β (decreasing vocabulary smoothing and hence increasing the effective number of clusters) beyond $\beta = 0.1$ actually hurts multi-prototype performance. The additional clusters do not provide more semantic content due to significant background similarity.

Finally, the DPMM multi-prototype and tiered clustering models allocate clusters based on the variance of the underlying data set. We observe a *negative* correlation (ρ=-0.33) between the number of clusters allocated by the DPMM and the number of word senses found in WordNet. This result is most likely due to our use of unigram context window features, which induce clustering based on thematic rather than syntactic differences. Investigating this issue is future work.

(**Future Work**) The word similarity experiments can be expanded by breaking pairs down further into highly homonymous and highly polysemous pairs, using e.g. WordNet to determine how closely related the senses are. With this data it would be interesting to validate the hypothesis that the percentage of features allocated to the background cluster is correlated with the degree of homonymy.

The basic tiered clustering can be extended with additional background tiers, allocating more expressivity to model background feature variation. This class of models covers the spectrum between a pure topic model (all background tiers) and a pure clustering model and may be reasonable when there is believed to be more background structure (e.g. when jointly modeling all verb arguments). Furthermore, it is straightforward to extend the model to a two-tier, two-clustering structure capable of additionally accounting for commonalities *between* arguments.

Applying more principled feature selection approaches to vector-space lexical semantics may yield more significant performance gains. Towards this end we are currently evaluating two classes of approaches for setting pruning parameters per-word instead of globally: (1) *subspace clustering*, i.e. unsupervised feature selection (e.g., Parsons et al., 2004) and (2) *multiple clustering*, i.e. finding feature partitions that lead to disparate clusterings (e.g., Shafto et al., 2006).

9 Conclusions

This paper introduced a simple probabilistic model of *tiered clustering* inspired by feature selective clustering that leverages feature exchangeability to allocate data features between a clustering model and shared component. The ability to model background variation, or shared structure, is shown to be beneficial for modeling words with high polysemy, yielding increased correlation with human similarity judgements modeling word relatedness and selectional preference. Furthermore, the tiered clustering model is shown to significantly outperform related models, yielding qualitatively more precise clusters.

Acknowledgments

Thanks to Yinon Bentor and Bryan Silverthorn for many illuminating discussions. This work was supported by an NSF Graduate Research Fellowship to the first author, and a Google Research Award.

A Collapsed Gibbs Sampler

In order to sample efficiently from this model, we leverage the Chinese Restaurant Process representation of the DP (cf., Aldous, 1985), introducing a per-word-occurrence cluster indicator c_d. Word occurrence features are then drawn from a combination of a single cluster component indicated by c_d and the background topic.

By exploiting conjugacy, the latent variables $\boldsymbol{\theta}$, $\boldsymbol{\phi}$ and η_d can be integrated out, yielding an efficient

[6] E.g., Padó et al. (2007) report ρ=0.515 on the same data.

collapsed Gibbs sampler. The likelihood of word occurrence d is given by

$$P(\mathbf{w}_d|\mathbf{z}, c_d, \boldsymbol{\phi}) = \\ \prod_i P(w_{i,d}|\boldsymbol{\phi}_{c_d})^{\delta(z_{d,i}=0)} P(w_{i,d}|\boldsymbol{\phi}_{\text{noise}})^{\delta(z_{d,i}=1)}.$$

Hence, this model can be viewed as a two-topic variant of LDA with the addition of a per-word-occurrence (i.e. document) cluster indicator.[7] The update rule for the latent tier indicator $\mathbf{z}$ is similar to the update rule for 2-topic LDA, with the background component as the first topic and the second topic being determined by the per-word-occurrence cluster indicator $\boldsymbol{c}$.

We can efficiently approximate $p(\mathbf{z}|\mathbf{w})$ via Gibbs sampling, which requires the complete conditional posteriors for all $z_{i,d}$. These are

$$P(z_{i,d} = t|\mathbf{z}_{-(i,d)}, \mathbf{w}, \alpha, \beta) = \\ \frac{n_t^{(w_{i,d})} + \beta}{\sum_w (n_t^{(w)} + \beta)} \frac{n_t^{(d)} + \alpha}{\sum_j (n_j^{(d)} + \alpha)}.$$

where $\mathbf{z}_{-(i,d)}$ is shorthand for the set $\mathbf{z} - \{z_{i,d}\}$, $n_t^{(w)}$ is the number of occurrences of word w in topic t not counting $w_{i,d}$ and $n_t^{(d)}$ is the number of features in occurrence d assigned to topic t, not counting $w_{i,d}$.

Likewise sampling the cluster indicators conditioned on the data $p(\boldsymbol{c}_d|\mathbf{w}, \boldsymbol{c}_{-d}, \alpha, \eta)$ decomposes into the DP posterior over cluster assignments and the cluster-conditional Multinomial-Dirichlet word-occurrence likelihood $p(\boldsymbol{c}_d|\mathbf{w}, \boldsymbol{c}_{-d}, \alpha, \eta) = p(\boldsymbol{c}_d|\boldsymbol{c}_{-d}, \eta) p(\mathbf{w}_d|\mathbf{w}_{-d}, \boldsymbol{c}, \mathbf{z}, \alpha)$ given by

$$P(c_d = k_{\text{old}}|\boldsymbol{c}_{-d}, \alpha, \eta) \propto \\ \underbrace{\left(\frac{m_k^{(-d)}}{m_{\blacksquare}^{(-d)} + \eta}\right)}_{p(\boldsymbol{c}_d|\boldsymbol{c}_{-d}, \eta)} \underbrace{\frac{C(\alpha + \overrightarrow{\boldsymbol{n}}_k^{(-d)} + \overrightarrow{\boldsymbol{n}}_{\blacksquare}^{(d)}))}{C(\alpha + \overrightarrow{\boldsymbol{n}}_k^{(-d)})}}_{p(\mathbf{w}_d|\mathbf{w}_{-d}, \boldsymbol{c}, \mathbf{z}, \alpha)}$$

$$P(c_d = k_{\text{new}}|\boldsymbol{c}_{-d}, \alpha, \eta) \propto \frac{\eta}{m_{\blacksquare}^{(-d)} + \eta} \frac{C(\alpha + \overrightarrow{\boldsymbol{n}}_{\blacksquare}^{(d)})}{C(\alpha)}$$

where $m_k^{(-d)}$ is the number of occurrences assigned to k not including d, $\overrightarrow{\boldsymbol{n}}_k^{(d)}$ is the vector of counts of words from occurrence $\mathbf{w}_d$ assigned to cluster k (i.e. words with $\mathbf{z}_{i,d} = 0$) and $C(\cdot)$ is the normalizing constant for the Dirichlet $C(\boldsymbol{a}) = \Gamma(\sum_{j=1}^m a_j)^{-1} \prod_{j=1}^m \Gamma(a_j)$ operating over vectors of counts $\boldsymbol{a}$.

[7] Effectively, the tiered clustering model is a special case of the *nested* Chinese Restaurant Process with the tree depth fixed to two (Blei et al., 2003).

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and Wordnet-based approaches. In *Proc. of NAACL-HLT-09*, pages 19–27.

David J. Aldous. 1985. Exchangeability and related topics. In *École d'été de probabilités de Saint-Flour, XIII—1983*, volume 1117, pages 1–198. Springer, Berlin.

David Blei, Thomas Griffiths, Michael Jordan, and Joshua Tenenbaum. 2003. Hierarchical topic models and the nested Chinese restaurant process. In *Proc. NIPS-2003*.

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

James Richard Curran. 2004. *From Distributional to Semantic Similarity*. Ph.D. thesis, University of Edinburgh. College of Science.

Katrin Erk and Sebastian Pado. 2008. A structured vector space model for word meaning in context. In *Proceedings of EMNLP 2008*.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database and Some of its Applications*. MIT Press.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: the concept revisited. In *Proc. of WWW 2001*.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *Proc. of IJCAI-07*, pages 1606–1611.

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

James Gorman and James R. Curran. 2006. Scaling distributional similarity to large corpora. In *Proc. of ACL 2006*.

Thomas L. Griffiths, Mark Steyvers, and Joshua B. Tenenbaum. 2007. Topics in semantic representation. *Psychological Review*, 114:2007.

Amaç Herdağdelen and Marco Baroni. 2009. Bagpack: A general framework to represent semantic relations. In *Proc. of GEMS 2009*.

Donald Hindle and Mats Rooth. 1991. Structural ambiguity and lexical relations. In *Proc. of ACL 1991*.

Thomas Landauer and Susan 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.

Martin H. C. Law, Anil K. Jain, and Mário A. T. Figueiredo. 2002. Feature selection in mixture-based clustering. In *Proc. of NIPS 2002*.

Will Lowe. 2001. Towards a theory of semantic space. In *Proceedings of the 23rd Annual Meeting of the Cognitive Science Society*, pages 576–581.

Xiaojuan Ma, Jordan Boyd-Graber, Sonya S. Nikolova, and Perry Cook. 2009. Speaking through pictures: Images vs. icons. In *ACM Conference on Computers and Accessibility*.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Diana McCarthy and John Carroll. 2003. Disambiguating nouns, verbs, and adjectives using automatically acquired selectional preferences. *Computational Linguistics*, 29(4):639–654.

George A. Miller and Walter G. Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Sebastian Padó, Ulrike Padó, and Katrin Erk. 2007. Flexible, corpus-based modelling of human plausibility judgements. In *Proc. of EMNLP 2007*.

Ulrike Padó. 2007. *The Integration of Syntax and Semantic Plausibility in a Wide-Coverage Model of Sentence Processing*. Ph.D. thesis, Saarland University, Saarbrücken.

Patrick Pantel, Rahul Bhagat, Timothy Chklovski, and Eduard Hovy. 2007. ISP: Learning inferential selectional preferences. In *In Proceedings of NAACL 2007*.

Patrick Andre Pantel. 2003. *Clustering by committee*. Ph.D. thesis, Edmonton, Alta., Canada.

Lance Parsons, Ehtesham Haque, and Huan Liu. 2004. Subspace clustering for high dimensional data: A review. *SIGKDD Explor. Newsl.*, 6(1).

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of English words. In *Proc. of ACL 1993*.

Carl E. Rasmussen. 2000. The infinite Gaussian mixture model. In *Advances in Neural Information Processing Systems*. MIT Press.

Joseph Reisinger and Raymond Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Proc. of NAACL 2010*.

Philip Resnik. 1997. Selectional preference and sense disambiguation. In *Proceedings of ACL SIGLEX Workshop on Tagging Text with Lexical Semantics*, pages 52–57. ACL.

Adam N. Sanborn, Thomas L. Griffiths, and Daniel J. Navarro. 2006. A more rational model of categorization. In *Proceedings of the 28th Annual Conference of the Cognitive Science Society*.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123.

Patrick Shafto, Charles Kemp, Vikash Mansinghka, Matthew Gordon, and Joshua B. Tenenbaum. 2006. Learning cross-cutting systems of categories. In *Proc. CogSci 2006*.

Rion Snow, Daniel Jurafsky, and Andrew Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proc. of ACL 2006*.

Peter D. Turney. 2006. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416.

Benjamin Van Durme and Marius Paşca. 2008. Finding cars, goddesses and enzymes: Parametrizable acquisition of labeled instances for open-domain information extraction. In *Proc. of AAAI 2008*.

Nianwen Xue, Jinying Chen, and Martha Palmer. 2006. Aligning features with sense distinction dimensions. In *Proc. of COLING/ACL 2006*.

Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space

Marco Baroni and **Roberto Zamparelli**
Center for Mind/Brain Sciences, University of Trento
Rovereto (TN), Italy
{marco.baroni,roberto.zamparelli}@unitn.it

Abstract

We propose an approach to adjective-noun composition (AN) for corpus-based distributional semantics that, building on insights from theoretical linguistics, represents nouns as vectors and adjectives as data-induced (linear) functions (encoded as matrices) over nominal vectors. Our model significantly outperforms the rivals on the task of reconstructing AN vectors not seen in training. A small post-hoc analysis further suggests that, when the model-generated AN vector is not similar to the corpus-observed AN vector, this is due to anomalies in the latter. We show moreover that our approach provides two novel ways to represent adjective meanings, alternative to its representation via corpus-based co-occurrence vectors, both outperforming the latter in an adjective clustering task.

1 Introduction

An influential approach for representing the meaning of a word in NLP is to treat it as a vector that codes the pattern of co-occurrence of that word with other expressions in a large corpus of language (Sahlgren, 2006; Turney and Pantel, 2010). This approach to semantics (sometimes called *distributional semantics*) naturally captures word clustering, scales well to large lexicons and doesn't require words to be manually disambiguated (Schütze, 1997). However, until recently it has been limited to the level of content words (nouns, adjectives, verbs), and it hasn't tackled in a general way *compositionality* (Frege, 1892; Partee, 2004), that crucial property of natural language which allows speakers to derive the meaning of a complex linguistic constituent from the meaning of its immediate syntactic subconstituents.

Formal semantics (FS), the research program stemming from Montague (1970b; 1973), has opposite strengths and weaknesses. Its core semantic notion is the sentence, not the word; at the lexical level, it focuses on the meaning of function words; one of its main goals is to formulate recursive compositional rules that derive the quantificational properties of complex sentences and their antecedent-pronoun dependencies.

Given its focus on quantification, FS treats the meanings of nouns and verbs as pure *extensions*: nouns and (intransitive) verbs are properties, and thus denote sets of individuals. Adjectives are also often assumed to denote properties: in this view red_{adj} would be the set of 'entities which are red', $plastic_{adj}$, the set of 'objects made of plastic', and so forth. In the simplest case, the meaning of an attributive adjective-noun (AN) constituent can be obtained as the intersection of the adjective and noun extensions A∩N:

[red car] = $\{\ldots$ *red objects* $\ldots\} \cap \{\ldots$ *cars* $\ldots\}$

However, the intersective method of combination is well-known to fail in many cases (Kamp, 1975; Montague, 1970a; Siegel, 1976): for instance, a *fake gun* is not a *gun*. Even for *red*, the manner in which the color combines with a noun will be different in *red Ferrari* (the outside), *red watermelon* (the inside), *red traffic light* (the signal). These problems have prompted a more flexible FS representation for attributive adjectives — functions from the meaning of a noun onto the meaning of a modified noun (Montague, 1970a). This mapping could now be sensitive to the particular noun the adjective receives, and it does not need to return a subset of the

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1183–1193,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

original noun denotation (as in the case of *fake N*). However, FS has nothing to say on how these functions should be constructed.

In the last few years there have been attempts to build compositional models that use distributional semantic representations as inputs (see Section 2 below), most of them focusing on the combination of a verb and its arguments. This paper addresses instead the combination of nouns and attributive adjectives. This case was chosen as an interesting testbed because it has the property of recursivity (it applies in *black dog*, but also in *large black dog*, etc.), and because very frequent adjectives such as *different* are at the border between content and function words. Following the insight of FS, we treat attributive adjectives as *functions* over noun meanings; however, noun meanings are vectors, not sets, and the functions are learnt from corpus-based noun-AN vector pairs.

Original contribution We propose and evaluate a new method to derive distributional representations for ANs, where an adjective is a linear function from a vector (the noun representation) to another vector (the AN representation). The linear map for a specific adjective is learnt, using linear regression, from pairs of noun and AN vectors extracted from a corpus.

Outline Distributional approaches to compositionality are shortly reviewed in Section 2. In Section 3, we introduce our proposal. The experimental setting is described in Section 4. Section 5 provides some empirical justification for using corpus-harvested AN vectors as the target of our function learning and evaluation benchmark. In Section 6, we show that our model outperforms other approaches at the task of approximating such vectors for unseen ANs. In Section 7, we discuss how adjectival meaning can be represented in our model and evaluate this representation in an adjective clustering task. Section 8 concludes by sketching directions for further work.

2 Related work

The literature on compositionality in vector-based semantics encompasses various related topics, some of them not of direct interest here, such as how to encode word order information in context vectors (Jones and Mewhort, 2007; Sahlgren et al., 2008) or sophisticated composition methods based on tensor products, quantum logic, etc., that have not yet been empirically tested on large-scale corpus-based semantic space tasks (Clark and Pulman, 2007; Rudolph and Giesbrecht, 2010; Smolensky, 1990; Widdows, 2008). Closer to our current purposes is the general framework for vector composition proposed by Mitchell and Lapata (2008), subsuming various earlier proposals. Given two vectors $\mathbf{u}$ and $\mathbf{v}$, they identify two general classes of composition models, (linear) additive models:

$$\mathbf{p} = \mathbf{A}\mathbf{u} + \mathbf{B}\mathbf{v} \tag{1}$$

where $\mathbf{A}$ and $\mathbf{B}$ are weight matrices, and multiplicative models:

$$\mathbf{p} = \mathbf{C}\mathbf{u}\mathbf{v}$$

where $\mathbf{C}$ is a weight tensor projecting the $\mathbf{uv}$ tensor product onto the space of $\mathbf{p}$. Mitchell and Lapata derive two simplified models from these general forms. Their simplified additive model $\mathbf{p} = \alpha\mathbf{u} + \beta\mathbf{v}$ was a common approach to composition in the earlier literature, typically with the scalar weights set to 1 or to normalizing constants (Foltz et al., 1998; Kintsch, 2001; Landauer and Dumais, 1997). Mitchell and Lapata also consider a constrained version of the multiplicative approach that reduces to component-wise multiplication, where the i-th component of the composed vector is given by: $p_i = u_i v_i$. The simplified additive model produces a sort of (statistical) union of features, whereas component-wise multiplication has an intersective effect. They also evaluate a weighted combination of the simplified additive and multiplicative functions. The best results on the task of paraphrasing noun-verb combinations with ambiguous verbs (*sales slump* is more like *declining* than *slouching*) are obtained using the multiplicative approach, and by weighted combination of addition and multiplication (we do not test model combinations in our current experiments). The multiplicative approach also performs best (but only by a small margin) in a later application to language modeling (Mitchell and Lapata, 2009). Erk and Padó (2008; 2009) adopt the same formalism but focus on the nature of input vectors, suggesting that when a verb is composed with a noun, the noun component is given by an average of verbs that the noun is typically object of (along similar lines, Kintsch (2001) also focused on composite input vectors, within an additive framework). Again, the multiplicative model works best in Erk and Padó's experiments.

The above-mentioned researchers do not exploit corpus evidence about the $\mathbf{p}$ vectors that result from composition, despite the fact that it is straightforward (at least for short constructions) to extract direct distributional evidence about the composite items from the corpus (just collect co-occurrence information for the composite item from windows around the contexts in which it occurs). The main innovation of Guevara (2010), who focuses on adjective-noun combinations (AN), is to use the co-occurrence vectors of observed ANs to train a supervised composition model (we became aware of Guevara's approach after we had developed our own model, that also exploits observed ANs for training). Guevara adopts the full additive composition form from Equation (1) and he estimates the $\mathbf{A}$ and $\mathbf{B}$ weights using partial least squares regression. The training data are pairs of adjective-noun vector concatenations, as input, and corpus-derived AN vectors, as output. Guevara compares his model to the simplified additive and multiplicative models of Mitchell and Lapata. Observed ANs are nearer, in the space of observed and predicted test set ANs, to the ANs generated by his model than to those from the alternative approaches. The additive model, on the other hand, is best in terms of shared neighbor count between observed and predicted ANs.

In our empirical tests, we compare our approach to the simplified additive and multiplicative models of Mitchell and Lapata (the former with normalization constants as scalar weights) as well as to Guevara's approach.

3 Adjectives as linear maps

As discussed in the introduction, we will take adjectives in attributive position to be functions from one noun meaning to another. To start simple, we assume here that adjectives in the attributive position (AN) are linear functions from n-dimensional (noun) vectors onto n-dimensional vectors, an operation that can be expressed as multiplication of the input noun column vector by a $n \times n$ matrix, that is our representation for the adjective (in the language of linear algebra, an adjective is an endomorphic linear map in noun space). In the framework of Mitchell and Lapata, our approach derives from the additive form in Equation (1) with the matrix multiplying the adjective vector (say, $\mathbf{A}$) set to $\mathbf{0}$:

$$\mathbf{p} = \mathbf{B}\mathbf{v}$$

where $\mathbf{p}$ is the observed AN vector, $\mathbf{B}$ the weight matrix representing the adjective at hand, and $\mathbf{v}$ a noun vector. In our approach, the weight matrix $\mathbf{B}$ is specific to a single adjective – as we will see in Section 7 below, it is our representation of the meaning of the adjective.

Like Guevara, we estimate the values in the weight matrix by partial least squares regression. In our case, the independent variables for the regression equations are the dimensions of the corpus-based vectors of the component nouns, whereas the AN vectors provide the dependent variables. Unlike Guevara, (i) we train separate models for each adjective (we learn adjective-specific functions, whereas Guevara learns a generic "AN-slot" function) and, consequently, (ii) corpus-harvested adjective vectors play no role for us (their values would be constant across the training input vectors).

A few considerations are in order. First, although we use a supervised learning method (least squares regression), we do not need hand-annotated data, since the target AN vectors are automatically collected from the corpus just like vectors for single words are. Thus, there is no extra "external knowledge" cost with respect to unsupervised approaches. Second, our approach rests on the assumption that the corpus-derived AN vectors are interesting objects that should constitute the target of what a composition process tries to approximate. We provide preliminary empirical support for this assumption in Section 5 below. Third, we have some reasonable hope that our functions can capture to a certain extent the polysemous nature of adjectives: we could learn, for example, a *green* matrix with large positive weights mapping from noun features that pertain to concrete objects to color dimensions of the output vector (*green chair*), as well as large positive weights from features characterizing certain classes of abstract concepts to political/social dimensions in the output (*green initiative*). Somewhat optimistically, we hope that *chair* will have near-0 values on the relevant abstract dimensions, like *initiative* on the concrete features, and thus the weights will not interfere. We do not evaluate this claim specifically, but our quantitative evaluation in Section 6 shows that our approach does best with high frequency, highly ambiguous adjectives. Fourth, the approach is naturally syntax-sensitive, since we train it on observed data for a specific syntactic position: we would train separate linear models for, say, the same adjective in attributive (AN) and predicative (N is A) position. As a matter of fact, the current model is too syntax-sensitive and does not capture similarities across different constructions. Finally, although adjective representations are not directly harvested from corpora, we can still meaningfully compare adjectives to each other or other words by using their estimated matrix, or an average vector for the ANs that contain them: both options are tested in Section 7 below.

4 Experimental setup

4.1 Corpus

We built a large corpus by concatenating the Web-derived ukWaC corpus (`http://wacky.sslmit.unibo.it/`), a mid-2009 dump of the English Wikipedia (`http://en.wikipedia.org`) and the British National Corpus (`http://www.natcorp.ox.ac.uk/`). This **concatenated corpus**, tokenized, POS-tagged and lemmatized with the TreeTagger (Schmid, 1995), contains about 2.83 billion tokens (excluding punctuation, digits, etc.). The ukWaC and Wikipedia sections can be freely downloaded, with full annotation, from the ukWaC site.

We performed some of the list extraction and checking operations we are about to describe on a more manageable data-set obtained by selecting the first 100M tokens of ukWaC; we refer to this subset as the **sample corpus** below.

4.2 Vocabulary

We could in principle limit ourselves to collecting vectors for the ANs to be analyzed (the *AN test set*) and their components. However, to make the analysis more challenging and interesting, we populate the semantic space where we will look at the behaviour of the ANs with a large number of adjectives and nouns, as well as further ANs not in the test set. We refer to the overall list of items we build semantic vectors for as the *extended vocabulary*. We use a subset of the extended vocabulary containing only nouns and adjectives (the *core vocabulary*) for feature selection and dimensionality reduction, so that we do not implicitly bias the structure of the semantic space by our choice of ANs.

To construct the **AN test set**, we first selected 36 adjectives across various classes: size (*big, great, huge, large, major, small, little*), denominal (*American, European, national, mental, historical, electronic*), colors (*white, black, red, green*) positive evaluation (*nice, excellent, important, appropriate*), temporal (*old, recent, new, young, current*), modal (*necessary, possible*), plus some common abstract antonymous pairs (*difficult, easy, good, bad, special, general, different, common*). We were careful to include intersective cases such as *electronic* as well as non-intersective adjectives that are almost function words (the modals, *different*, etc.). We extracted all nouns that occurred at least 300 times in post-adjectival position in the sample corpus, excluding some extremely frequent temporal and measure expressions such as *time* and *range*, for a total of 1,420 distinct nouns. By crossing the selected adjectives and nouns, we constructed a test set containing 26,440 ANs, all attested in the sample corpus (734 ANs per adjective on average, ranging from 1,337 for *new* to 202 for *mental*).

The **core vocabulary** contains the top 8K most frequent noun lemmas and top 4K adjective lemmas from the concatenated corpus (excluding the top 50 most frequent nouns and adjectives). The **extended vocabulary** contains this core plus (i) the 26,440 test ANs, (ii) the 16 adjectives and 43 nouns that are components of these ANs and that are not in the core set, and (iii) 2,500 more ANs randomly sampled from those that are attested in the sample corpus, have a noun from the same list used for the test set ANs, and an adjective that occurred at least 5K times in the sample corpus. In total, the extended vocabulary contains 40,999 entries: 8,043 nouns, 4,016 adjectives and 28,940 ANs.

4.3 Semantic space construction

Full co-occurrence matrix The 10K lemmas (nouns, adjectives or verbs) that co-occur with the largest number of items in the core vocabulary constitute the dimensions (columns) of our co-occurrence matrix. Using the concatenated corpus, we extract sentence-internal co-occurrence counts of all the items in the extended vocabulary with the 10K dimension words. We then transform the raw counts into Local Mutual Information (LMI) scores (LMI is an association measure that closely approximates the Log-Likelihood Ratio, see Evert (2005)).

Dimensionality reduction Since, for each test set adjective, we need to estimate a regression model for each dimension, we want a compact space with relatively few, dense dimensions. A natural way to do this is to apply the Singular Value Decomposition (SVD) to the co-occurrence matrix, and represent the items of interest with their coordinates in the space spanned by the first n right singular vectors. Applying SVD is independently justified because, besides mitigating the dimensionality problem, it often improves the quality of the semantic space (Landauer and Dumais, 1997; Rapp, 2003; Schütze, 1997). To avoid bias in favour of dimensions that capture variance in the test set ANs, we applied SVD to the core vocabulary subset of the co-occurrence matrix (containing only adjective and noun rows). The core 12K×10K matrix was reduced using SVD to a 12K×300 matrix. The other row vectors of the full co-occurrence matrix (including the ANs) were projected onto the same reduced space by multiplying them by a matrix containing the first n right singular vectors as columns. Merging the items used to compute the SVD and those projected onto the resulting space, we obtain a 40,999×300 matrix representing 8,043 nouns, 4,016 adjectives and 28,940 ANs. This reduced matrix constitutes a realistically sized semantic space, that also contains many items that are not part of our test set, but will be potential neighbors of the observed and predicted test ANs in the experiments to follow. The quality of the SVD reduction itself was independently validated on a standard similarity judgment data-set (Rubenstein and Goodenough, 1965), obtaining similar (and state-of-the-art-range) Pearson correlations of vector cosines and human judgments in both the original ($r = .70$) and reduced ($r = .72$) spaces.

There are several parameters involved in constructing a semantic space (choice of full and reduced dimensions, co-occurrence span, weighting method). Since our current focus is on alternative composition methods evaluated on a shared semantic space, exploring parameters pertaining to the construction of the semantic space is not one of our priorities, although we cannot of course exclude that the nature of the underlying semantic space affects different composition methods differently.

4.4 Composition methods

In the proposed **adjective-specific linear map** (*alm*) method, an AN is generated by multiplying an adjective weight matrix with a noun (column) vector. The j weights in the i-th row of the matrix are the coefficients of a linear regression predicting the values of the i-th dimension of the AN vector as a linear combination of the j dimensions of the component noun. The linear regression coefficients are estimated separately for each of the 36 tested adjectives from the corpus-observed noun-AN pairs containing that adjective (observed adjective vectors are not used). Since we are working in the 300-dimensional right singular vector space, for each adjective we have 300 regression problems with 300 independent variables, and the training data (the noun-AN pairs available for each test set adjective) range from about 200 to more than 1K items. We estimate the coefficients using (multivariate) partial least squares regression (PLSR) as implemented in the R `pls` package (Mevik and Wehrens, 2007). With respect to standard least squares estimation, this technique is more robust against over-training by effectively using a smaller number of orthogonal "latent" variables as predictors (Hastie et al., 2009, Section 3.4), and it exploits the multivariate nature of the problem (different regressions for each AN vector dimension to be predicted) when determining the latent dimensions. The number of latent variables to be used in the core regression are a free parameter of PLSR. For efficiency reasons, we did not optimize it. We picked instead 50 latent variables, by the rule-of-thumb reasoning that for any adjective we can use at least 200 noun-AN pairs for training, and the independent-variable-to-training-item ratio will thus never be above 1/4. We adopt a leave-one-out training regime, so that each target AN is generated by an adjective matrix that was estimated from all the other ANs with the same adjective, minus the target.

We use PLSR with 50 latent variables also for our re-implementation of Guevara's (2010) **single linear map** (*slm*) approach, in which a single regression matrix is estimated for all ANs across adjectives. The training data in this case are given by the concatenation of the observed adjective and noun vectors (600 independent variables) coupled with the corresponding AN vectors (300 dependent variables). For each target AN, we randomly sample 2,000 other adjective-noun-AN tuples for training (with larger training sets we run into memory problems), and use the resulting coefficient matrix to generate the AN vector from the concatenated target adjective and noun vectors.

Additive AN vectors (*add* method) are obtained by summing the corresponding adjective and noun vectors after normalizing them (non-normalized addition was also tried, but it did not work nearly as well as the normalized variant). **Multiplicative** vectors (*mult* method) were obtained by component-wise multiplication of the adjective and noun vectors (normalization does not matter here since it amounts to multiplying the composite vector by a scalar, and the cosine similarity measure we use is scale-invariant). Finally, the *adj* and *noun* baselines use the **adjective** and **noun** vectors, respectively, as surrogates of the AN vector.

For the *add*, *mult*, *adj* and *noun* methods, we ran the tests of Section 6 not only in the SVD-reduced space, but also in the original 10K-dimensional co-occurrence space. Only the *mult* method achieved better performance in the original space. We conjecture that this is because the SVD dimensions can have negative values, leading to counter-intuitive results with component-wise multiplication (multiplying large opposite-sign values results in large negative values). We tried to alleviate this problem by assigning a 0 to composite dimensions where the two input vectors had different signs. The resulting performance was better but still below that of *mult* in original space. Thus, in Section 6 we report *mult* results from the full co-occurrence matrix; reduced space results for all other methods.

5 Study 1: ANs in semantic space

The actual distribution of ANs in the corpus, as recorded by their co-occurrence vectors, is fundamental to what we are doing. Our method relies on the hypothesis that the semantics of AN composition does not depend on the independent distribution of adjectives themselves, but on how adjectives transform the distribution of nouns, as evidenced by observed pairs of noun-AN vectors. Moreover, coherently with this view, our evaluation below will be based on how closely the models approximate the observed vectors of unseen ANs.

That our goal in modeling composition should be to approximate the vectors of observed ANs is in a sense almost trivial. Whether we synthesize an AN for generation or decoding purposes, we would want the synthetic AN to look as much as possible like a real AN in its natural usage contexts, and co-occurrence vectors of observed ANs are a summary of their usage in actual linguistic contexts. However, it might be the case that the specific resources we used for our vector construction procedure are not appropriate, so that the specific observed AN vectors we extract are not reliable (e.g., they are so sparse in the original space as to be uninformative, or they are strictly tied to the domains of the input corpora). We provide here some preliminary qualitative evidence that this is in general not the case, by tapping into our own intuitions on where ANs should be located in semantic space, and thus on how sensible their neighbors are.

First, we computed centroids from normalized SVD space vectors of all the ANs that share the same adjective (e.g., the normalized vectors of *American adult*, *American menu*, etc., summed to construct the *American N* centroid). We looked at the nearest neighbors of these centroids in semantic space among the 41K items (adjectives, nouns and ANs) in our extended vocabulary (here and in all experiments below, similarity is quantified by the cosine of the angle between two vectors). As illustrated for a random sample of 9 centroids in Table 1 (but applying to the remaining 27 adjectives as well), centroids are positioned in intuitively reasonable areas of the space, typically near the adjective itself or the corresponding noun (the noun *green* near *green N)*, prototypical ANs for that adjective (*black face*), elements related to the definition of the adjective (*mental activity*, *historical event*, *green colour*, *quick* and *little cost* for *easy N*), and so on.

American N	*black N*	*easy N*
Am. representative Am. territory Am. source	black face black hand black (n)	easy start quick little cost
green N	*historical N*	*mental N*
green (n) red road green colour	historical hist. event hist. content	mental activity mental experience mental energy
necessary N	*nice N*	*young N*
necessary necessary degree sufficient	nice good bit nice break	youthful young doctor young staff

Table 1: Nearest 3 neighbors of centroids of ANs that share the same adjective.

How about the neighbors of specific ANs? Table 2 reports the nearest 3 neighbors of 9 randomly selected ANs involving different adjectives (we inspected a larger random set, coming to similar conclusions to the ones emerging from this table).

bad luck	*electronic communication*	*historical map*
bad bad weekend good spirit	elec. storage elec. transmission purpose	topographical atlas hist. material
important route	*nice girl*	*little war*
important transport important road major road	good girl big girl guy	great war major war small war
red cover	*special collection*	*young husband*
black cover hardback red label	general collection small collection archives	small son small daughter mistress

Table 2: Nearest 3 neighbors of specific ANs.

The nearest neighbors of the corpus-based AN vectors in Table 2 make in general intuitive sense. Importantly, the neighbors pick up the composite meaning rather than that of the adjective or noun alone. For example, *cover* is an ambiguous word, but the *hardback* neighbor relates to its "front of a book" meaning that is the most natural one in combination with *red*. Similarly, it makes more sense that a *young husband* (rather than an old one) would have *small sons* and *daughters* (not to mention the *mistress*!).

We realize that the evidence presented here is of a very preliminary and intuitive nature. Indeed, we will argue in the next section that there are cases in which the corpus-derived AN vector might not be a good approximation to our semantic intuitions about the AN, and a model-composed AN vector is a better semantic surrogate. One of the most important avenues for further work will be to come to a better characterization of the behaviour of corpus-observed ANs, where they work and where the don't. Still, the neighbors of average and AN-specific vectors of Tables 1 and 2 suggest that, for the bulk of ANs, such corpus-based co-occurrence vectors are semantically reasonable.

6 Study 2: Predicting AN vectors

Having tentatively established that the sort of vectors we can harvest for ANs by directly collecting their corpus co-occurrences are reasonable representations of their composite meaning, we move on to the core question of whether it is possible to reconstruct the vector for an unobserved AN from information about its components. We use nearness to the corpus-observed vectors of held-out ANs as a very direct way to evaluate the quality of model-generated ANs, since we just saw that the observed ANs look reasonable (but see the caveats at the end of this section). We leave it to further work to assess the quality of the generated ANs in an applied setting, for example adapting Mitchell and Lapata's paraphrasing task to ANs. Since the observed vectors look like plausible representations of composite meaning, we expect that the closer the model-generated vectors are to the observed ones, the better they should also perform in any task that requires access to the composite meaning, and thus that the results of the current evaluation should correlate with applied performance.

More in detail, we evaluate here the composition methods (and the *adjective* and *noun* baselines) by computing, for each of them, the cosine of the test set AN vectors they generate (the "predicted" ANs) with the 41K vectors representing our extended vocabulary in semantic space, and looking at the position of the corresponding observed ANs (that were not used for training, in the supervised approaches) in the cosine-ranked lists. The lower the rank, the better the approximation. For efficiency reasons, we flatten out the ranks after the top 1,000 neighbors.

The results are summarized in Table 3 by the median and the other quartiles, calculated across all 26,440 ANs in the test set. These measures (unlike mean and variance) are not affected by the cut-off after 1K neighbors. To put the reported results into perspective, a model with a first quartile rank of 999 does very significantly better than chance (the binomial probability of $1/4$ or more of 26,440 trials being successful with $\pi = 0.024$ is virtually 0, where the latter quantity is the probability of an observed AN being at rank 999 or lower according to a geometric distribution with $\pi = 1/40999$).

method	*25%*	*median*	*75%*
alm	17	170	≥1K
add	27	257	≥1K
noun	72	448	≥1K
mult	279	≥1K	≥1K
slm	629	≥1K	≥1K
adj	≥1K	≥1K	≥1K

Table 3: Quartile ranks of observed ANs in cosine-ranked lists of predicted AN neighbors.

Our proposed method, *alm*, emerges as the best approach. The difference with the second best model, *add* (the only other model that does better than the non-trivial baseline of using the component noun vector as a surrogate for AN), is highly statistically significant (Wilcoxon signed rank test, $p < 0.00001$). If we randomly downsample the AN set to keep an equal number of ANs per adjective (200), the difference is still significant with p below the same threshold, indicating that the general result is not due to a better performance of *alm* on a few common adjectives.[1]

Among the alternative models, the fact that the performance of *add* is decidedly better than that of *mult* is remarkable, since earlier studies found that multiplicative models are, in general, better than additive ones in compositionality tasks (see Section 2 above). This might depend on the nature of AN composition, but there are also more technical issues at hand: (i) we are not sure that previous studies normalized before summing like we did, and (ii) the multiplicative model, as discussed in Section 4, does not benefit from SVD reduction. The single linear mapping model (*slm*) proposed by Guevara (2010) is doing even worse than the multiplicative method, suggesting that a single set of weights does not provide enough flexibility to model a variety of adjective transformations successfully. This is at odds with Guevara's experiment in which *slm* outperformed *mult* and *add* on the task of ranking predicted ANs with respect to a target observed AN. Besides various differences in task definition and model implementation, Guevara trained his model on ANs that include a wide variety of adjectives, whereas our training data were limited to ANs containing one of our 36 test set adjectives. Future work should re-evalute the performance of Guevara's approach in our task, but under his training regime.

Looking now at the *alm* results in more detail, the best median ranks are obtained for very frequent adjectives. The top ones are *new* (median rank: 34), *great* (79), *American* (82), *large* (82) and *different* (97). There is a high inverse correlation between median rank and adjective frequency (Spearman's $\rho = -0.56$). Although from a statistical perspective it is expected that we get better results where we have more data, from a linguistic point of view it is interesting that *alm* works best with extremely frequent, highly polysemous adjectives like *new*, *large* and *different*, that border on function words – a domain where distributional semantics has generally not been tested.

Although, in relative terms and considering the difficulty of the task, *alm* performs well, it is still far from perfect – for 27% *alm*-predicted ANs, the observed vector is not even in the top 1K neighbor set! A qualitative look at some of the most problematic examples indicates however that a good proportion of them might actually not be instances where our model got the AN vector wrong, but cases of anomalous observed ANs. The left side of Table 4 compares the nearest neighbors (excluding each other) of the observed and *alm*-predicted vectors in 10 ran-

[1] The semantic space in which we rank the observed ANs with respect to their predicted counterparts also contain the observed vectors of nouns and ANs that were used to train *alm*. We do not see how this should affect performance, but we nevertheless repeated the evaluation leaving out, for each AN, the observed items used in training, and we obtained the same results reported in the main text (same ordering of method performance, and very significant difference between *alm* and *add*).

SIMILAR			DISSIMILAR		
adj N	*obs. neighbor*	*pred. neighbor*	*adj N*	*obs. neighbor*	*pred. neighbor*
common understanding	common approach	common vision	American affair	Am. development	Am. policy
different authority	diff. objective	diff. description	current dimension	left (a)	current element
different partner	diff. organisation	diff. department	good complaint	current complaint	good beginning
general question	general issue	*same*	great field	excellent field	gr. distribution
historical introduction	hist. background	*same*	historical thing	different today	hist. reality
necessary qualification	nec. experience	*same*	important summer	summer	big holiday
new actor	new cast	*same*	large pass	historical region	large dimension
recent request	recent enquiry	*same*	special something	little animal	special thing
small drop	droplet	drop	white profile	chrome (n)	white show
young engineer	young designer	y. engineering	young photo	important song	young image

Table 4: Left: nearest neighbors of observed and *alm*-predicted ANs (excluding each other) for a random set of ANs where rank of observed w.r.t. predicted is 1. Right: nearest neighbors of predicted and observed ANs for random set where rank of observed w.r.t. predicted is $\geq 1K$.

domly selected cases where the observed AN is the nearest neighbor of the predicted one. Here, the ANs themselves make sense, and the (often shared) neighbors are also sensible (*recent enquiry* for *recent request*, *common approach* and *common vision* for *common understanding*, etc.). Moving to the right, we see 10 random examples of ANs where the observed AN was at least 999 neighbors apart from the *alm* prediction. First, we notice some ANs that are difficult to interpret out-of-context (*important summer*, *white profile*, *young photo*, *large pass*, ...). Second, at least subjectively, we find that in many cases the nearest neighbor of predicted AN is actually more sensible than that of observed AN: *current element* (vs. *left*) for *current dimension*, *historical reality* (vs. *different today*) for *historical thing*, *special thing* (vs. *little animal*) for *special something*, *young image* (vs. *important song*) for *young photo*. In the other cases, the predicted AN neighbor is at least not obviously worse than the observed AN neighbor.

There is a high inverse correlation between the frequency of occurrence of an AN and the rank of the observed AN with respect to the predicted one ($\rho = -0.48$), suggesting that our model is worse at approximating the observed vectors of rare forms, that might, in turn, be those for which the corpus-based representation is less reliable. In these cases, dissimilarities between observed and expected vectors, rather than signaling problems with the model, might indicate that the predicted vector, based on a composition function learned from many examples, is *better* than the one directly extracted from the corpus. The examples in the right panel of Table 4 bring some preliminary support to this hypothesis, to be systematically explored in future work.

7 Study 3: Comparing adjectives

If adjectives are functions, and not corpus-derived vectors, is it still possible to compare them meaningfully? We explore two ways to accomplish this in our framework: one is to represent adjectives by the average of the AN vectors that contain them (the centroid vectors whose neighbors are illustrated in Table 1 above), and the other to compare them based on the 300×300 weight matrices we estimate from noun-AN pairs (we unfold these matrices into 90K-dimensional vectors). We compare the quality of these representations to that of the standard approach in distributional semantics, i.e., representing the adjectives directly with their corpus co-occurrence profile vectors (in our case, projected onto the SVD-reduced space).

We evaluate performance on the task of clustering those 19 adjectives in our set that can be relatively straightforwardly categorized into general classes comprising a minimum of 4 items. The test set built according to these criteria contains 4 classes: color (*white*, *black*, *red*, *green*), positive evaluation (*nice*, *excellent*, *important*, *major*, *appropriate*), time (*recent*, *new*, *current*, *old*, *young*), and size (*big*, *huge*, *little*, *small*, *large*). We cluster with the CLUTO toolkit (Karypis, 2003), using the *repeated bisections with global optimization* method, accepting all of CLUTO's default values for this choice. Cluster quality is evaluated by percentage *purity* (Zhao and Karypis, 2003). If n_r^i is the number of items from the *i*-th true (gold standard) class assigned to the *r*-th cluster, n is the total number of items and k the number of clusters, then: $Purity = \frac{1}{n}\sum_{r=1}^{k} \max_i(n_r^i)$. We calculate empirical 95% confidence intervals around purity by a heuristic bootstrap procedure based on 10K resamplings of the data set (Efron and Tibshirani, 1994). The random baseline distribution is obtained by 10K random assignments of adjectives to the clusters, under the constraint that no cluster is empty.

Table 5 shows that all methods are significantly better than chance. Our two "indirect" representations achieve similar performance, and they are (slightly) better than the traditional method based on adjective co-occurrence vectors. We conclude that, although our approach does not provide a direct encoding of adjective meaning in terms of such independently collected vectors, it does have meaningful ways to represent their semantic properties.

input	*purity*
matrix	73.7 (68.4-94.7)
centroid	73.7 (63.2-94.7)
vector	68.4 (63.2-89.5)
random	45.9 (36.8-57.9)

Table 5: Percentage purity in adjective clustering with bootstrapped 95% confidence intervals.

8 Conclusion

The work we reported constitutes an encouraging start for our approach to modeling (AN) composition. We suggested, along the way, various directions for further studies. We consider the following issues to be the most pressing ones.

We currently train each adjective-specific model separately: We should explore hierarchical modeling approaches that exploit similarities across adjectives (and possibly syntactic constructions) to estimate better models.

Evaluation-wise, the differences between observed and predicted ANs must be analyzed more extensively, to support the claim that, when their vectors differ, model-based prediction improves on the observed vector. Evaluation in a more applied task should also be pursued – in particular, we will design a paraphrasing task similar to the one proposed by Mitchell and Lapata to evaluate noun-verb constructions.

Since we do not collect vectors for the "functor" component of a composition process (for AN constructions, the adjective), our approach naturally extends to processes that involve bound morphemes, such as affixation, where we would not need to collect independent co-occurrence information for the affixes. For example, to account for *re-* prefixation we do not need to collect a *re-* vector (required by all other approaches to composition), but simply vectors for a set of *V*/*reV* pairs, where both members of the pairs are words (e.g., *consider*/*reconsider*).

Our approach can also deal, out-of-the-box, with recursive constructions (*sad little red hat*), and can be easily extended to more abstract constructions, such as *determiner N* (mapping *dog* to *the/a/one dog*). Still, we need to design a good testing scenario to evaluate the quality of such model-generated constructions.

Ultimately, we want to compose larger and larger constituents, up to full sentences. It remains to be seen if the approach we proposed will scale up to such challenges.

Acknowledgments

We thank Gemma Boleda, Emilano Guevara, Alessandro Lenci, Louise McNally and the anonymous reviewers for useful information, advice and comments.

References

S. Clark and S. Pulman. 2007. Combining symbolic and distributional models of meaning. In *Proceedings of the First Symposium on Quantum Interaction*, pages 52–55.

B. Efron and R. Tibshirani. 1994. *An Introduction to the Bootstrap*. Chapman and Hall, Boca Raton, FL.

K. Erk and S. Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of EMNLP*, pages 897–906.

K. Erk and S. Padó. 2009. Paraphrase assessment in structured vector space: Exploring parameters and datasets. In *Proceedings of the EACL GEMS Workshop*, pages 57–65.

S. Evert. 2005. *The Statistics of Word Cooccurrences*. Dissertation, Stuttgart University.

P. Foltz, W. Kintsch, and Th. Landauer. 1998. The measurement of textual coherence with Latent Semantic Analysis. *Discourse Processes*, 25:285–307.

G. Frege. 1892. Über sinn und bedeutung. *Zeitschrift fuer Philosophie un philosophische Kritik*, 100.

E. Guevara. 2010. A regression model of adjective-noun compositionality in distributional semantics. In *Proceedings of the ACL GEMS Workshop*, pages 33–37.

T. Hastie, R. Tibshirani, and J. Friedman. 2009. *The Elements of Statistical Learning, 2nd ed.* Springer, New York.

M. Jones and D. Mewhort. 2007. Representing word meaning and order information in a composite holographic lexicon. *Psychological Review*, 114:1–37.

H. Kamp. 1975. Two theories about adjectives. In E. Keenan, editor, *Formal Semantics of Natural Language*, pages 123–155. Cambridge University Press.

G. Karypis. 2003. CLUTO: A clustering toolkit. Technical Report 02-017, University of Minnesota Department of Computer Science.

W. Kintsch. 2001. Predication. *Cognitive Science*, 25(2):173–202.

Th. 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(2):211–240.

B. Mevik and R. Wehrens. 2007. The pls package: Principal component and partial least squares regression in R. *Journal of Statistical Software*, 18(2).

J. Mitchell and M. Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL*, pages 236–244.

J. Mitchell and M. Lapata. 2009. Language models based on semantic composition. In *Proceedings of EMNLP*, pages 430–439.

R. Montague. 1970a. English as a formal language. In B. Visentini, editor, *Linguaggi nella Società e nella Tecnica*, pages 189–224. Edizioni di Comunità, Milan. Reprinted in Thomason (1974).

R. Montague. 1970b. Universal grammar. *Theoria*, 36:373–398. Reprinted in Thomason (1974).

R. Montague. 1973. The proper treatment of quantification in English. In K.J.J. Hintikka, editor, *Approaches to Natural Language*, pages 221–242. Reidel, Dordrecht. Reprinted in Thomason (1974).

B. Partee. 2004. Compositionality. In *Compositionality in Formal Semantics: Selected Papers by Barbara H. Partee*. Blackwell, Oxford.

R. Rapp. 2003. Word sense discovery based on sense descriptor dissimilarity. In *Proceedings of the MT Summit*, pages 315–322.

H. Rubenstein and J. Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

S. Rudolph and E. Giesbrecht. 2010. Compositional matrix-space models of language. In *Proceedings of ACL*.

M. Sahlgren, A. Holst, and P. Kanerva. 2008. Permutations as a means to encode order in word space. In *Proceedings of CogSci*, pages 1300–1305.

M. Sahlgren. 2006. *The Word-Space Model*. Dissertation, Stockholm University.

H. Schmid. 1995. Improvements in part-of-speech tagging with an application to German. In *Proceedings of the EACL-SIGDAT Workshop*.

H. Schütze. 1997. *Ambiguity Resolution in Natural Language Learning*. CSLI, Stanford, CA.

M. Siegel. 1976. *Capturing the Adjective*. Ph.D. thesis, University of Massachusetts at Amherst.

P. Smolensky. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist networks. *Artificial Intelligence*, 46:159–216.

R. H. Thomason, editor. 1974. *Formal Philosophy: Selected Papers of Richard Montague*. Yale University Press, New York.

P. Turney and P. Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

D. Widdows. 2008. Semantic vector products: Some initial investigations. In *Proceedings of the Second Symposium on Quantum Interaction*, Oxford.

Y. Zhao and G. Karypis. 2003. Criterion functions for document clustering: Experiments and analysis. Technical Report 01-40, University of Minnesota Department of Computer Science.

Practical Linguistic Steganography using Contextual Synonym Substitution and Vertex Colour Coding

Ching-Yun Chang
University of Cambridge
Computer Laboratory
Ching-Yun.Chang@cl.cam.ac.uk

Stephen Clark
University of Cambridge
Computer Laboratory
Stephen.Clark@cl.cam.ac.uk

Abstract

Linguistic Steganography is concerned with hiding information in natural language text. One of the major transformations used in Linguistic Steganography is synonym substitution. However, few existing studies have studied the practical application of this approach. In this paper we propose two improvements to the use of synonym substitution for encoding hidden bits of information. First, we use the Web 1T Google n-gram corpus for checking the applicability of a synonym in context, and we evaluate this method using data from the SemEval lexical substitution task. Second, we address the problem that arises from words with more than one sense, which creates a potential ambiguity in terms of which bits are encoded by a particular word. We develop a novel method in which words are the vertices in a graph, synonyms are linked by edges, and the bits assigned to a word are determined by a vertex colouring algorithm. This method ensures that each word encodes a unique sequence of bits, without cutting out large number of synonyms, and thus maintaining a reasonable embedding capacity.

1 Introduction

Steganography is concerned with hiding information in a cover medium, in order to facilitate covert communication, such that the presence of the information is imperceptible to a user (human or computer). Much of the existing research in steganography has used images as cover media; however, given the ubiquitous nature of electronic text, interest is growing in using natural language as the cover medium. Linguistic Steganography—lying at the intersection of Computational Linguistics and Computer Security—is concerned with making changes to a cover text in order to embed information, in such a way that the changes do not result in ungrammatical or unnatural text.

A related area is natural language watermarking, in which changes are made to a text in order to identify it, for example for copyright purposes. An interesting watermarking application is "traitor tracing", in which documents are changed in order to embed individual watermarks. These marks can then be used to later identify particular documents, for example if a set of documents—identical except for the changes used to embed the watermarks— have been sent to a group of individuals, and one of the documents has been leaked to a newspaper.

In terms of security, a linguistic stegosystem should impose minimum embedding distortion to the cover text so that the resulting stegotext in which a message is camouflaged is inconspicuous, resulting in high *imperceptibility*. In addition, since steganography aims at covert communication, a linguistic stegosystem should allow sufficient embedding capacity, known as the *payload*. There is a fundamental tradeoff between imperceptibility and payload, since any attempt to embed more information via changes to the cover text increases the chance of introducing anomalies into the text and therefore raising the suspicion of an observer.

A linguistic transformation is required to embed information. Transformations studied in previous work include lexical substitution (Chapman and Davida, 1997; Bolshakov, 2004; Taskiran et al., 2006; Topkara et al., 2006b), phrase paraphrasing (Chang and Clark, 2010), sentence structure manipulations (Atallah et al., 2001a; Atallah et al., 2001b;

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1194–1203,
MIT, Massachusetts, USA, 9-11 October 2010.

Liu et al., 2005; Meral et al., 2007; Murphy, 2001; Murphy and Vogel, 2007; Topkara et al., 2006a) and semantic transformations (Atallah et al., 2002; Vybornova and Macq, 2007). Many of these transformations require some sophisticated NLP tools; for example, in order to perform semantic transformations on text, word sense disambiguation, semantic role parsing and anaphora resolution tools may be required. However, the current state-of-the-art in language technology is arguably not good enough for secure linguistic steganography based on sophisticated semantic transformations, and the level of robustness required to perform practical experiments has only just become available. Hence many existing linguistic stegosystems are proof-of-concept implementations with little practical evaluation of the imperceptibility or payload.

1.1 Synonym substitution

Synonym substitution is a relatively straightforward linguistic steganography method. It substitutes selected words with the same part of speech (PoS) synonyms, and does not involve operating on the sentence structure so the modification can be guaranteed to be grammatical. Another advantage of this method is that many languages are profuse in synonyms, and so there is a rich source of information carriers compared with other text transformations.

There are two practical difficulties associated with hiding bits using synonym subsitution. The first is that words can have more than one sense. In terms of WordNet (Fellbaum, 1998), which is the electronic dictionary we use, words can appear in more than one synset. This is a problem because a word may be assigned different bit strings in the different synsets, and the receiver does not know which of the senses to use, and hence does not know which hidden bit string to recover. Our solution to this problem is a novel vertex colouring method which ensures that words are always assigned the same bit string, even when they appear in different synsets.

The second problem is that many synonyms are only applicable in certain contexts. For example, the words in the WordNet synset {*bridge*, *span*} share the meaning of "a structure that allows people or vehicles to cross an obstacle such as a river or canal or railway etc.". However, *bridge* and *span* cannot be substutited for each other in the sentence "suspension bridges are typically ranked by the length of their main span", and doing so would likely raise the suspicion of an observer due to the resulting anomaly in the text.

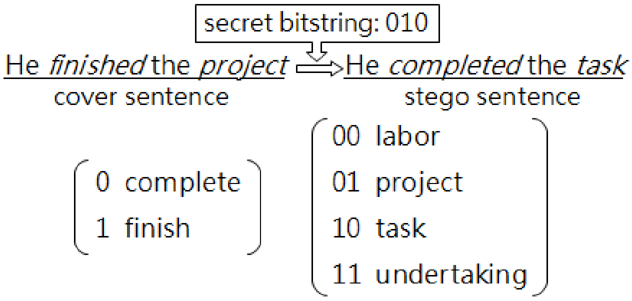


Figure 1: An example of the basic algorithm

Our solution to this problem is to perform a contextual check which utilises the Web 1T n-gram Google n-gram corpus.[1] We evaluate the method using the data from the English Lexical Substitution task for SemEval-2007.[2] The resulting precision of our lexical substitution system can be seen as an indirect measure of the imperceptibility of the stegosystem, whereas the recall can be seen as an indirect measure of the payload.

The paper is organised so that the contextual check is described first, and this is evaluated independently of the steganographic application. Then the vertex colouring method is presented, and finally we show how the contextual check can be integrated with the vertex colouring coding method to give a complete stegsosystem. For readers unfamiliar with linguistic steganogaphy, Section 2 has some examples of how bits can be hidden using textual transformations. Also, Chang and Clark (2010) is a recent NLP paper which describes the general linguistic steganography framework.

2 Related Work

In the original work on linguistic steganography in the late 1990s, Winstein proposed an information hiding algorithm using a block coding method to encode synonyms, so that the selection of a word from a synset directly associates with part of the secret bitstring (Bergmair, 2007). Figure 1 illustrates the embedding procedure of this approach. In this example, the bitstring to be embedded is 010, which

[1] www.ldc.upenn.edu/Catalog/docs/LDC2006T13/readme.txt

[2] http://www.dianamccarthy.co.uk/task10index.html

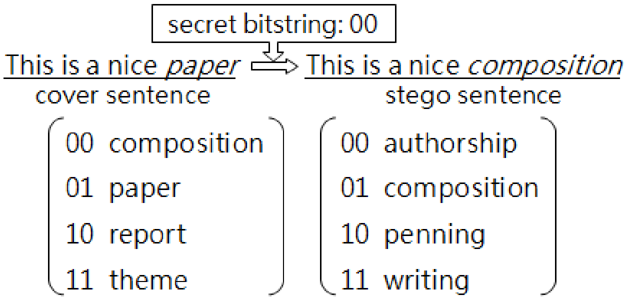


Figure 2: An example of applying the basic algorithm to overlapping synsets

can be divided into two codewords, 0 and 10, and the information carriers in the cover text are the words *finished* and *project*. According to the encoding dictionary, *complete* represents 0, and *task* represents 10; hence these words are chosen and replace the original words in the cover text (with suitable suffixation). The stego sentence "He completed the task" is then sent to the receiver. In order to recover the message, the receiver only needs a copy of the encoding dictionary, and the decoding algorithm simply reverses the process used to encode the hidden bits. Note that the receiver does not need the original cover text to recover the information.

This algorithm requires synonym sets to be disjoint; i.e. no word may appear in more than one synonym set, since overlapping synsets may cause ambiguities during the decoding stage. Figure 2 shows what happens when the basic algorithm is applied to two overlapping synonym sets. As can be seen from the example, *composition* is represented by two different codewords and thus the secret bitstring cannot be reliably recovered, since the receiver does not know the original cover word or the sense of the word. In order to solve this problem, we propose a novel coding method based on vertex colouring, described in Section 4.

In addition to the basic algorithm, Winstein proposed the T-Lex system using synonym substitution as the text transformation. In order to solve the problem of words appearing in more than one synonym set, Winstein defines *interchangeable words* as words that belong to the same synsets, and only uses these words for substitution. Any words that are not interchangeable are discarded and not available for carrying information. The advantage in this approach is that interchangeable words always receive the same codeword. The disadvantage is that many synonyms need to be discarded in order to achieve this property. Winstein calculates that only 30% of WordNet can be used in such a system.

Another stegosystem based on synonym substitution was proposed by Bolshakov (2004). In order to ensure both sender and receiver use the same synsets, Bolshakov applied *transitive closure* to overlapping synsets to avoid the decoding ambiguity. Applying transitive closure leads to a merger of all the overlapping synsets into one set which is then seen as the synset of a target word. Consider the overlapping synsets in Figure 2 as an example. After applying transitive closure, the resulting set is {'authorship', 'composition', 'paper', 'penning', 'report', 'theme', 'writing'}.

Bolshakov (2004) also uses a method to determine whether a substitution is applicable in context, using a collocation-based test. Finally, the collocationally verified synonyms are encoded by using the block coding method. This is similar to our use of the Google n-gram data to check for contextual applicability.

The disadvantage of Bolshakov's system is that all words in a synonym transitive closure chain need to be considered, which can lead to very large sets of synonyms, and many which are not synonymous with the original target word. In contrast, our proposed method operates on the original synonym sets without extending them unnecessarily.

3 Proposed Method and Experiments

3.1 Resources

We use WordNet (Fellbaum, 1998) to provide sets of synonyms (synsets) for nouns, verbs, adjectives and adverbs. Since the purpose of using WordNet is to find possible substitutes for a target word, those synsets containing only one entry are not useful and are ignored by our stegosystem. In addition, our stegosystem only takes single word substitution into consideration in order to avoid the confusion of finding information-carrying words during the decoding phase. For example, if the cover word 'complete' is replaced by 'all over', the receiver would not know whether the secret message is embedded in the word 'over' or the phrase 'all over'. Table 1 shows the statistics of synsets used in our stegosystem.

	noun	verb	adj	adv
# of synsets	16,079	4,529	6,655	964
# of entries	30,933	6,495	14,151	2,025
average set size	2.56	2.79	2.72	2.51
max set size	25	16	21	8

Table 1: Statistics of synsets used in our stegosystem

For the contextual check we use the Google Web 1T 5-gram Corpus (Brants and Franz, 2006) which contains counts for n-grams from unigrams through to five-grams obtained from over 1 trillion word tokens of English Web text. The corpus has been used for many tasks such as spelling correction (Islam and Inkpen, 2009; Carlson et al., 2008) and multi-word expression classification (Kummerfeld and Curran, 2008). Moreover, for the SemEval-2007 English Lexical Substitution Task, which is similar to our substitution task, six out of ten participating teams utilised the Web 1T corpus.

3.2 Synonym Checking Method

In order to measure the degree of acceptability in a substitution, the proposed filter calculates a substitution score for a synonym by using the observed frequency counts in the Web n-gram corpus. The method first extracts contextual n-grams around the synonym and queries the n-gram frequency counts from the corpus. For each n, the total count f_n is calculated by summing up individual n-gram frequencies, for every contextual n-gram containing the target word. We define a *count function* $Count(w) = \sum_{n=2}^{5} log(f_n)$ where $log(0)$ is defined as zero. If $Count(w) = 0$, we assume the word w is unrelated to the context and therefore is eliminated from consideration. We then find the maximum $Count(w)$ called max from the remaining words. The main purpose of having max is to score each word relative to the most likely synonym in the group, so even in less frequent contexts which lead to smaller frequency counts, the score of each synonym can still indicate the degree of feasibility. The substitution score is defined as $Score(w) = Count(w) \div max$. The hypothesis is that a word with a high score is more suitable for the context, and we apply a threshold so that synonyms having a score lower than the threshold are discarded.

f_2=525,856	high *pole*	3,544
	pole .	522,312
f_3=554	very high *pole*	84
	high *pole* .	470
f_4=72	a very high *pole*	72
	very high *pole* .	0
f_5=0	not a very high *pole*	0
	a very high *pole* .	0
Count('*pole*')=log(f_2)+log(f_3)+log(f_4)+log(f_5)=23		
Score('*pole*')=Count('*pole*')/max=0.44>0.37		

Figure 3: An example of using the proposed synonym checking method

Figure 3 demonstrates an example of calculating the substitution score for the synonym 'pole' given the cover sentence "This is not a very high *bar*." First of all, various contextual n-grams are extracted from the sentence and the Web n-gram corpus is consulted to obtain their frequency counts. $Count(\text{'pole'})$ is then calculated using the n-gram frequencies. Suppose the threshold is 0.37, and the max score is 52. Since $Count(\text{'pole'})$ is greater than zero and the substitution score $Score(\text{'pole'})$ is 0.44, the word 'pole' is determined as acceptable for this context (even though it may not be, depending on the meaning of 'bar' in this case).

3.3 Evaluation Data

In order to evaluate the proposed synonym checking method, we need some data to test whether our method can pick out acceptable substitutions. The English Lexical Substitution task for SemEval-2007 has created human-annotated data for developing systems that can automatically find feasible substitutes given a target word in context. This data comprises 2010 sentences selected from the English Internet Corpus[3], and consists of 201 words: nouns, verbs, adjectives and adverbs each with ten sentences containing that word. The five annotators were asked to provide up to three substitutes for a target word in the context of a sentence, and were permitted to consult a dictionary or thesaurus of their choosing.

We use the sentences in this gold standard as the cover text in our experiments so that the substitutes provided by the annotators can be the positive data for evaluating the proposed synonym check-

[3] http://corpus.leeds.ac.uk/internet.html

	noun	verb	adj	adv
# of target words	59	54	57	35
# of sentences	570	527	558	349
# of positives	2,343	2,371	2,708	1,269
# of negatives	1,914	1,715	1,868	884

Table 2: Statistics of experimental data

ing methods. Since we only take into consideration the single word substitutions for the reason described earlier, multi-word substitutes are removed from the positive data. Moreover, we use WordNet as the source of providing candidate substitutes in our stegosystem, so if a human-provided substitute does not appear in any synsets of its target word in WordNet, there is no chance for our system to replace the target word with the substitute and therefore, the substitute can be eliminated. Table 2 presents the statistics of the positive data for our experiments.

Apart from positive data, we also need some negative data to test whether our method has the ability to filter out bad substitutions. Since the annotators were allowed to refer to a dictionary or thesaurus, we assume that annotators used WordNet as one of the reference resources while generating candidates. Hence we assume that, if a word in the correct synset for a target word is not in the set produced by the human annotators, then it is inappropriate for that context and a suitable negative example. This method is appropriate because our steganography system has to distinguish between good and bad synonyms from WordNet, given a particular context.

For the above reasons, we extract the negative data for our experiments by first matching positive substitutes of a target word to all the synsets that contain the target word in WordNet. The synset that includes the most positive substitutes is used to represent the meaning of the target word. If there is more than one synset containing the highest number of positives, all the synsets are taken into consideration. We then randomly select up to six single-word synonyms other than positive substitutes from the chosen synset(s) as negative instances of the target word. Figure 4 shows an example of automatically collected negative data from WordNet given a target word and its positive substitutes. The synset {‘remainder’, ‘balance’, ‘residual’, ‘residue’, ‘residuum’, ‘rest’} is selected for negative data collection since it contains one of the positives while the other synsets do not. We assume the selected synset represents the meaning of the original word, and those synonyms in the synset which are not annotated as positives must have a certain degree of mismatch to the context. Therefore, from this example, ‘balance’, ‘residue’, ‘residuum’ and ‘rest’ are extracted as negatives to test whether our synonym checking method can pick out bad substitutions from a set of words sharing similar or the same meaning.

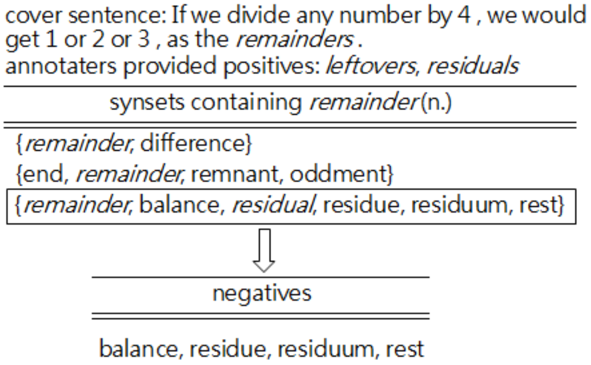

cover sentence: If we divide any number by 4 , we would get 1 or 2 or 3 , as the *remainders* .
annotaters provided positives: *leftovers, residuals*
synsets containing *remainder*(n.)
{*remainder*, difference}
{end, *remainder*, remnant, oddment}
{*remainder*, balance, *residual*, residue, residuum, rest}
negatives
balance, residue, residuum, rest

Figure 4: An example of automatic negative data

	noun	verb	adj	adv
# of true negatives	234	201	228	98
# of false negatives	9	20	28	16

Table 3: Annotation results for negative data

In order to examine whether the automatically collected instances are true negatives and hence form a useful test set, a sample of automatically generated negatives was selected for human evaluation. For each PoS one sentence of each different target word is selected, which results in roughly 13% of the collected negative data, and every negative substitute of the selected sentences was judged by the second author. As can be seen from the annotation results shown in Table 3, most of the instances are true negatives, and only a few cases are incorrectly chosen as false negatives. Since the main purpose of the data set is to test whether the proposed synonym checking method can guard against inappropriate synonym substitutions and be integrated in the stegosystem, it is reasonable to have a few false negatives in our experimental data. Also, it is more harmless to rule out a permissible substitu-

PoS	Acc%	P%	R%	F%	Threshold
noun	70.2	70.0	80.2	74.7	0.58
verb	68.1	69.7	79.5	74.3	0.56
adj	72.5	72.7	85.7	78.7	0.48
adv	73.7	76.4	80.1	78.2	0.54

Table 4: Performance of the synonym checking method

tion than including an inappropriate replacement for a stegosystem in terms of the security. Table 2 gives the statistics of the automatically collected negative data for our experiments.

Note that, although we use the data from the lexical substitution task, our task is different: the possible substitutions for a target word need to be fixed in advance for linguistic steganography (in order for the receiver to be able to recover the hidden bits), whereas for the lexical substitution task participants were asked to discover possible replacements.

3.4 Results

The performance of the proposed checking method is evaluated in terms of accuracy, precision, recall and balanced F-measure. Accuracy represents the percentage of correct judgements over all acceptable and unacceptable substitutions. Precision is the percentage of substitutions judged acceptable by the method which are determined to be suitable synonyms by the human judges. Recall is the percentage of substitutions determined to be feasible by the human annotators which are also judged acceptable by the method. The interpretation of the measures for a stegosystem is that a higher precision value implies a better security level since good substitutions are less likely to be seen as suspicious by the observer; whereas a larger recall value means a greater payload capacity since words are being substituted where possible and therefore embedding as much information as possible.

In order to derive sensible threshold values for each PoS, 5-fold cross-validation was implemented to conduct the experiments. For each fold, 80% of the data is used to find the threshold value which maximises the accuracy, and that threshold is then applied to the remaining 20% to get the final result. Table 4 gives the results for the synonym checking method and the average threshold values over the 5 folds. In addition, we are interested in the effect of various thresholds on the system performance. Figure 5 shows the precision and recall values with respect to different thresholds for each PoS. From the graphs we can clearly see the trade-off between precision and recall. Although a higher precision can be achieved by using a higher threshold value, for example noun's substitutions almost reach 90% precision with threshold equal to 0.9, the large drop in recall means many applicable synonyms are being eliminated. In other words, the trade-off between precision and recall implies the trade-off between imperceptibility and payload capacity for linguistic steganography. Therefore, the practical threshold setting would depend on how steganography users want to trade off imperceptibility for payload.

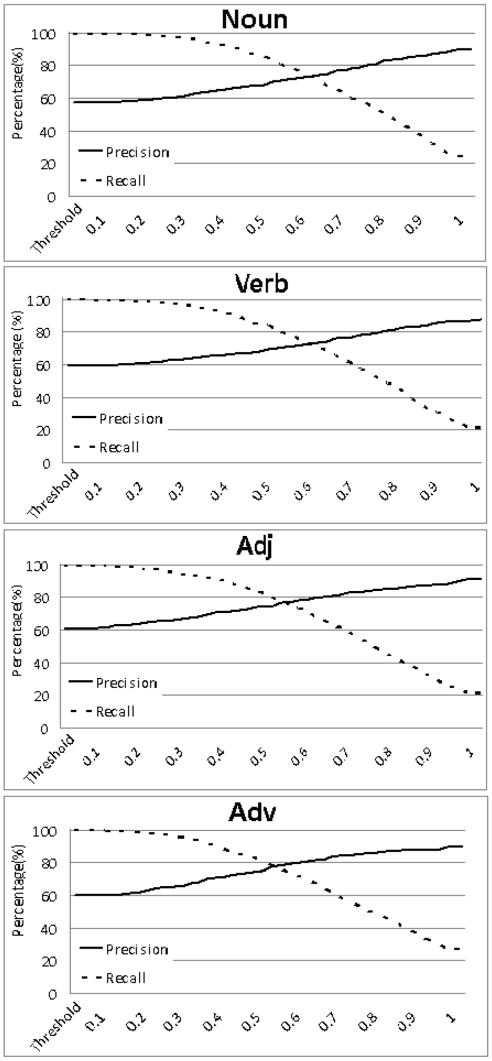


Figure 5: System performance under various thresholds

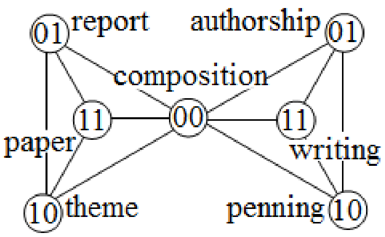


Figure 6: An example of coloured synonym graph

4 Proposed Stegosystem

4.1 The Vertex Coloring Coding Method

In this section, we propose a novel coding method based on vertex colouring by which each synonym is assigned a unique codeword so the usage of overlapping synsets is not problematic for data embedding and extracting. A vertex colouring is a labelling of the graph's vertices with colours subject to the condition that no two adjacent vertices share the same colour. The smallest number of colours required to colour a graph G is called its chromatic number $\chi(G)$, and a graph G having chromatic number $\chi(G) = k$ is called a k-chromatic graph. The main idea of the proposed coding method is to represent overlapping synsets as an undirected k-chromatic graph called a synonym graph which has a vertex for each word and an edge for every pair of words that share the same meaning. A synonym is then encoded by a codeword that represents the colour assigned by the vertex colouring of the synonym graph. Figure 6 shows the use of four different colours, represented by '00', '01', '10' and '11', to colour the 4-chromatic synonym graph of the two overlapping synsets in Figure 2. Now, the overlapped word 'composition' receives a unique codeword no matter which synset is considered, which means the replacement of 'paper' to 'composition' in Figure 2 will not cause an ambiguity since the receiver can apply the same coding method to derive identical codewords used by the sender.

99.6% of synsets in WordNet have size less than 8, which means most of the synsets cannot exhaust more than a 2-bit coding space (i.e. we can only encode at most 2 bits using a typical synset). Therefore, we restrict the chromatic number of a synonym graph G to $1 < \chi(G) \leq 4$, which implies the maximum size of a synset is 4. When $\chi(G) = 2$, each vertex is assigned a single-bit codeword either '0' or '1' as shown in Figure 7(a). When $\chi(G) = 3$, the overlapping set's size is either 2 or 3, which cannot exhaust the 2-bit coding space although codewords '00', '01' and '10' are initially assigned to each vertex. Therefore, only the most significant bits are used to represent the synonyms, which we call *codeword reduction*. After the codeword reduction, if a vertex has the same codeword, say '0', as all of its neighbors, the vertex's codeword must be changed to '1' so that the vertex would be able to accommodate either secret bit '0' or '1', which we call *codeword correction*. Figure 7(b) shows an example of the process of codeword reduction and codeword correction for $\chi(G) = 3$. For the case of $\chi(G) = 4$, codeword reduction is applied to those vertices that themselves or their neighboring vertices have no access to all the codewords '00', '01', '10' and '11'. For example, vertices a, b, c, e and f in Figure 7(c) meet the requirement of needing codeword reduction. The codeword correction process is then further applied to vertex f to rectify its accessibility.

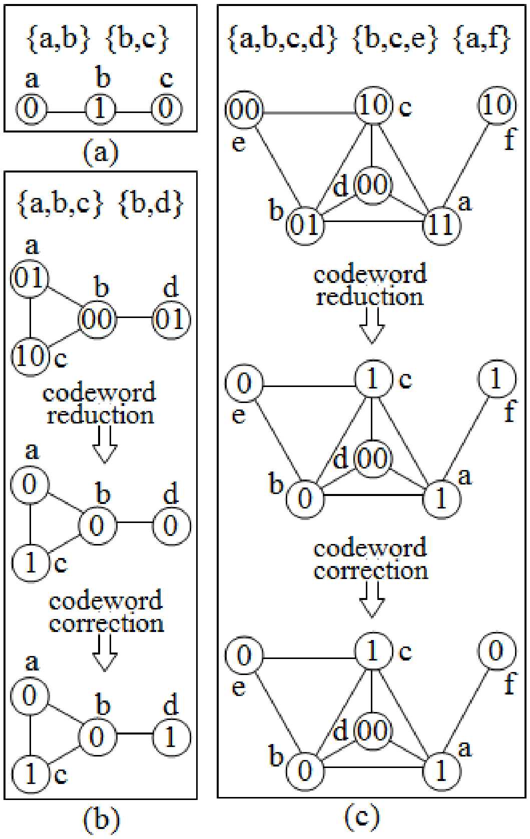


Figure 7: Examples of 2,3,4-chromatic synonym graphs

Figure 8 describes a greedy algorithm for constructing a coded synonym graph using at most 4 colours, given n synonyms w_1, w_2,$\ldots$, w_n in the overlapping synsets. Let us define a function $E(w_i, w_j)$ which returns an edge between w_i and w_j if w_i and w_j are in the same synset; otherwise returns false. Another function $C(w_i)$ returns the colour of the synonym w_i. The procedure loops through all the input synonyms. For each iteration, the procedure first finds available colours for the target synonym w_i. If there is no colour available, namely all the four colours have already been given to w_i's neighbors, w_i is randomly assigned one of the four colours; otherwise, w_i is assigned one of the available colours. After adding w_i to the graph G, the procedure checks whether adding an edge of w_i to graph G would violate the vertex colouring. After constructing the coloured graph, codeword reduction and codeword correction as previously described are applied to revise improper codewords.

4.2 Proposed Lexical Stegosystem

Figure 9 illustrates the framework of our lexical stegosystem. Note that we have preprocessed WordNet by excluding multi-word synonyms and single-entry synsets. A possible information carrier is first found in the cover sentence. We define a possible information carrier as a word in the cover sentence that belongs to at least one synset in WordNet. The synsets containing the target word, and all other synsets which can be reached via the synonym relation, are then extracted from WordNet (i.e. we build the connected component of WordNet which contains the target word according to the synonym relation). Words in these sets are then examined by the Google n-gram contextual checking method to eliminate inappropriate substitutions. If there is more than one word left and if words which pass the filter all belong to the same synset, the block coding method is used to encode the words; otherwise the vertex colouring coding is applied. Finally, according to the secret bitstring, the system selects the synonym that shares an edge with the target word and has as its codeword the longest potential match with the secret bitstring.

We use the connected component of WordNet containing the target word as a simple method to ensure that both sender and receiver colour-code the same graph. It is important to note, however, that the sender only considers the synonyms of the target word as potential substitutes; the connected component is only used to consistently assign the codes.

```
INPUT: a synonym list $w_1$, $w_2$,$\ldots$, $w_n$ and an
empty graph $G$
OUTPUT: a coded synonym graph $G$ using at
most four colours

FOR every synonym $w_i$ in the input list
    initialize four colours as available for $w_i$
    FOR every $w_j$ in graph $G$
        IF $E(w_i, w_j)$ THEN
            set $C(w_j)$ as unavailable
        END IF
    END FOR
    IF there is a colour available THEN
        assign one of the available colours
        to $w_i$
    ELSE
        assign one of the four colours to $w_i$
    END IF
    ADD $w_i$ to graph $G$
    FOR every $w_j$ in graph $G$
        IF $E(w_i, w_j)$ and $C(w_i)$ is not
        equal to $C(w_j)$ THEN
            ADD edge $E(w_i, w_j)$ to $G$
        END IF
    END FOR
END FOR
codeword reduction
codeword correction
OUTPUT graph $G$
```

Figure 8: Constructing a coloured synonym graph

For the decoding process, the receiver does not need the original text for extracting secret data. An information carrier can be found in the stegotext by referring to WordNet in which related synonyms are extracted. Those words in the related sets undergo the synonym checking method and then are encoded by either block coding or vertex colouring coding scheme depending on whether the remaining words are in the same synset. Finally, the secret bitstring is implicit in the codeword of the information carrier and therefore can be extracted.

We demonstrate how to embed secret bit 1 in the

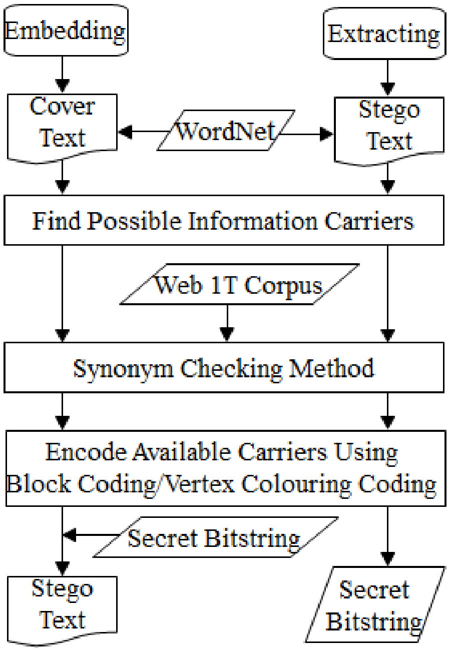


Figure 9: Framework of the proposed lexical stegosystem

sentence “it is a *shame* that we could not reach the next stage.” A possible information carrier ‘shame’ is first found in the sentence. Table 5 lists the related synsets extracted from WordNet. The score of each word calculated by the synonym checking method using the Web 1T Corpus is given as a subscript. Assume the threshold score is 0.27. The output of the synonym checking method is shown at the right side of Table 5. Since the remaining words do not belong to the same synset, the vertex colouring coding method is then used to encode the words. Figure 10(a) is the original synset graph in which each vertex is assigned one of the four colours; Figure 10(b) is the graph after applying codeword reduction. Although both ‘disgrace’ and ‘pity’ are encoded by ‘1’, ‘pity’ is chosen to replace the cover word since it has a higher score. Finally, the stegotext is generated, “it is a *pity* that we could not reach the next stage.”

As a rough guide to the potential payload with this approach, we estimate that, with a threshold of 0.5 for the contextual check, the payload would be slightly higher than 1 bit per newspaper sentence.

cover sentence:
It is a *shame* that we could not reach the next stage

original synsets	retained synsets
$\{commiseration_{.28}, pity_{.97}, ruth_{.13}, pathos_{.31}\}$	$\{commiseration, pity, pathos\}$
$\{pity_{.97}, shame_{1}\}$	$\{pity, shame\}$
$\{compassion_{.49}, pity_{.97}\}$	$\{compassion, pity\}$
$\{condolence_{.27}, commiseration_{.28}\}$	$\{commiseration\}$
$\{pathos_{.31}, poignancy_{.31}\}$	$\{pathos, poignancy\}$
$\{shame_{1}, disgrace_{.84}, ignominy_{.24}\}$	$\{shame, disgrace\}$
$\{compassion_{.49}, compassionateness_{0}\}$	$\{compassion\}$
$\{poignance_{.12}, poignancy_{.31}\}$	$\{poignancy\}$

Table 5: Synsets of ‘shame’ before and after applying the synonym checking method

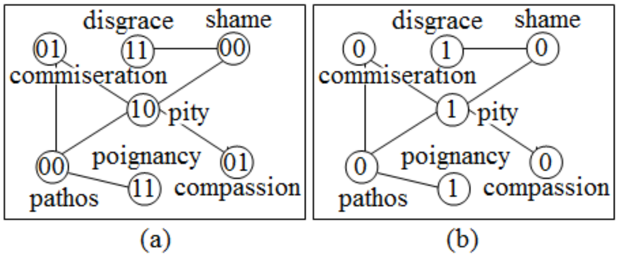


Figure 10: Synonym graph of ‘shame’

5 Conclusions

One of the contributions of this paper is to develop a novel lexical stegosystem based on vertex colouring coding which improves the data embedding capacity compared to existing systems. The vertex colouring coding method represents synonym substitution as a synonym graph so the relations between words can be clearly observed. In addition, an automatic system for checking synonym acceptability in context is integrated in our stegosystem to ensure information security. For future work, we would like to explore more linguistic transformations that can meet the requirements of linguistic steganography — retaining the meaning, grammaticality and style of the original text. In addition, it is crucial to have a full evaluation of the linguistic stegosystem in terms of imperceptibility and payload capacity so we can know how much data can be embedded before the cover text reaches its maximum distortion which is tolerated by a human judge.

References

Mikhail J. Atallah, Craig J. McDonough, Victor Raskin, and Sergei Nirenburg. 2001a. Natural language processing for information assurance and security: an overview and implementations. In *Proceedings of the 2000 workshop on New security paradigms*, pages 51–65, Ballycotton, County Cork, Ireland.

Mikhail J. Atallah, Victor Raskin, Michael C. Crogan, Christian Hempelmann, Florian Kerschbaum, Dina Mohamed, and Sanket Naik. 2001b. Natural language watermarking: design, analysis, and a proof-of-concept implementation. In *Proceedings of the 4th International Information Hiding Workshop*, volume 2137, pages 185–199, Pittsburgh, Pennsylvania.

Mikhail J. Atallah, Victor Raskin, Christian F. Hempelmann, Mercan Karahan, Umut Topkara, Katrina E. Triezenberg, and Radu Sion. 2002. Natural language watermarking and tamperproofing. In *Proceedings of the 5th International Information Hiding Workshop*, pages 196–212, Noordwijkerhout, The Netherlands.

Richard Bergmair. 2007. A comprehensive bibliography of linguistic steganography. In *Proceedings of the SPIE Conference on Security, Steganography, and Watermarking of Multimedia Contents*, volume 6505.

Igor A. Bolshakov. 2004. A method of linguistic steganography based on collocationally-verified synonym. In *Information Hiding: 6th International Workshop*, volume 3200, pages 180–191, Toronto, Canada.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram corpus version 1.1. Technical report, Google Research.

Andrew Carlson, Tom M. Mitchell, and Ian Fette. 2008. Data analysis project: Leveraging massive textual corpora using n-gram statistics. Technical report, School of Computer Science, Carnegie Mellon University.

Ching-Yun Chang and Stephen Clark. 2010. Linguistic steganography using automatically generated paraphrases. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 591–599, Los Angeles, California, June. Association for Computational Linguistics.

Mark Chapman and George I. Davida. 1997. Hiding the hidden: A software system for concealing ciphertext as innocuous text. In *Proceedings of the First International Conference on Information and Communication Security*, volume 1334, pages 335–345, Beijing.

Christiane Fellbaum. 1998. *WordNet: An electronic lexical database*. MIT Press, first edition.

Aminul Islam and Diana Inkpen. 2009. Real-word spelling correction using Google Web IT 3-grams. In *EMNLP '09: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1241–1249, Morristown, USA. Association for Computational Linguistics.

Jonathan K Kummerfeld and James R Curran. 2008. Classification of verb particle constructions with the Google Web 1T Corpus. In *Proceedings of the Australasian Language Technology Association Workshop 2008*, pages 55–63, Hobart, Australia, December.

Yuling Liu, Xingming Sun, and Yong Wu. 2005. A natural language watermarking based on Chinese syntax. In *Advances in Natural Computation*, volume 3612, pages 958–961, Changsha, China.

Diana McCarthy and Roberto Navigli. 2007. Semeval-2007 task 10: English lexical substitution task. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 48–53, Prague, Czech Republic.

Hasan M. Meral, Emre Sevinc, Ersin Unkar, Bulent Sankur, A. Sumru Ozsoy, and Tunga Gungor. 2007. Syntactic tools for text watermarking. In *Proceedings of the SPIE Conference on Security, Steganography, and Watermarking of Multimedia Contents*, volume 6505, San Jose, CA.

Brian Murphy and Carl Vogel. 2007. The syntax of concealment: reliable methods for plain text information hiding. In *Proceedings of the SPIE Conference on Security, Steganography, and Watermarking of Multimedia Contents*, volume 6505, San Jose, CA.

Brian Murphy. 2001. Syntactic information hiding in plain text. Master's thesis, Trinity College Dublin.

Cuneyt M. Taskiran, Mercan Topkara, and Edward J. Delp. 2006. Attacks on linguistic steganography systems using text analysis. In *Proceedings of the SPIE Conference on Security, Steganography, and Watermarking of Multimedia Contents*, volume 6072, pages 97–105, San Jose, CA.

Mercan Topkara, Umut Topkara, and Mikhail J. Atallah. 2006a. Words are not enough: sentence level natural language watermarking. In *Proceedings of the ACM Workshop on Content Protection and Security*, pages 37–46, Santa Barbara, CA.

Umut Topkara, Mercan Topkara, and Mikhail J. Atallah. 2006b. The hiding virtues of ambiguity: quantifiably resilient watermarking of natural language text through synonym substitutions. In *Proceedings of the 8th Workshop on Multimedia and Security*, pages 164–174, Geneva, Switzerland.

M. Olga Vybornova and Benoit Macq. 2007. A method of text watermarking using presuppositions. In *Proceedings of the SPIE Conference on Security, Steganography, and Watermarking of Multimedia Contents*, volume 6505, San Jose, CA.

Unsupervised Induction of Tree Substitution Grammars for Dependency Parsing

Phil Blunsom
Computing Laboratory
University of Oxford
Phil.Blunsom@comlab.ox.ac.uk

Trevor Cohn
Department of Computer Science
University of Sheffield
T.Cohn@dcs.shef.ac.uk

Abstract

Inducing a grammar directly from text is one of the oldest and most challenging tasks in Computational Linguistics. Significant progress has been made for inducing dependency grammars, however the models employed are overly simplistic, particularly in comparison to supervised parsing models. In this paper we present an approach to dependency grammar induction using tree substitution grammar which is capable of learning large dependency fragments and thereby better modelling the text. We define a hierarchical non-parametric Pitman-Yor Process prior which biases towards a small grammar with simple productions. This approach significantly improves the state-of-the-art, when measured by head attachment accuracy.

1 Introduction

Grammar induction is a central problem in Computational Linguistics, the aim of which is to induce linguistic structures from an unannotated text corpus. Despite considerable research effort this unsupervised problem remains largely unsolved, particularly for traditional phrase-structure parsing approaches (Clark, 2001; Klein and Manning, 2002). Phrase-structure parser induction is made difficult due to two types of ambiguity: the constituent structure and the constituent labels. In particular the constituent labels are highly ambiguous, firstly we don't know *a priori* how many there are, and secondly labels that appear high in a tree (e.g., an *S* category for a clause) rely on the correct inference of all the latent labels below them. However recent work on the induction of dependency grammars has proved more fruitful (Klein and Manning, 2004). Dependency grammars (Mel′čuk, 1988) should be easier to induce from text compared to phrase-structure grammars because the set of labels (heads) are directly observed as the words in the sentence.

Approaches to unsupervised grammar induction, both for phrase-structure and dependency grammars, have typically used very simplistic models (Clark, 2001; Klein and Manning, 2004), especially in comparison to supervised parsing models (Collins, 2003; Clark and Curran, 2004; McDonald, 2006). Simple models are attractive for grammar induction because they have a limited capacity to overfit, however they are incapable of modelling many known linguistic phenomena. We posit that more complex grammars could be used to better model the unsupervised task, provided that active measures are taken to prevent overfitting. In this paper we present an approach to dependency grammar induction using a tree-substitution grammar (TSG) with a Bayesian non-parametric prior. This allows the model to learn large dependency fragments to best describe the text, with the prior biasing the model towards fewer and smaller grammar productions.

We adopt the split-head construction (Eisner, 2000; Johnson, 2007) to map dependency parses to context free grammar (CFG) derivations, over which we apply a model of TSG induction (Cohn et al., 2009). The model uses a hierarchical Pitman-Yor process to encode a backoff path from TSG to CFG rules, and from lexicalised to unlexicalised rules. Our best lexicalised model achieves a head attachment accuracy of of 55.7% on Section 23 of the WSJ data set, which significantly improves over state-of-the-art and far exceeds an EM baseline (Klein and Manning, 2004) which obtains 35.9%.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1204–1213,
MIT, Massachusetts, USA, 9-11 October 2010.

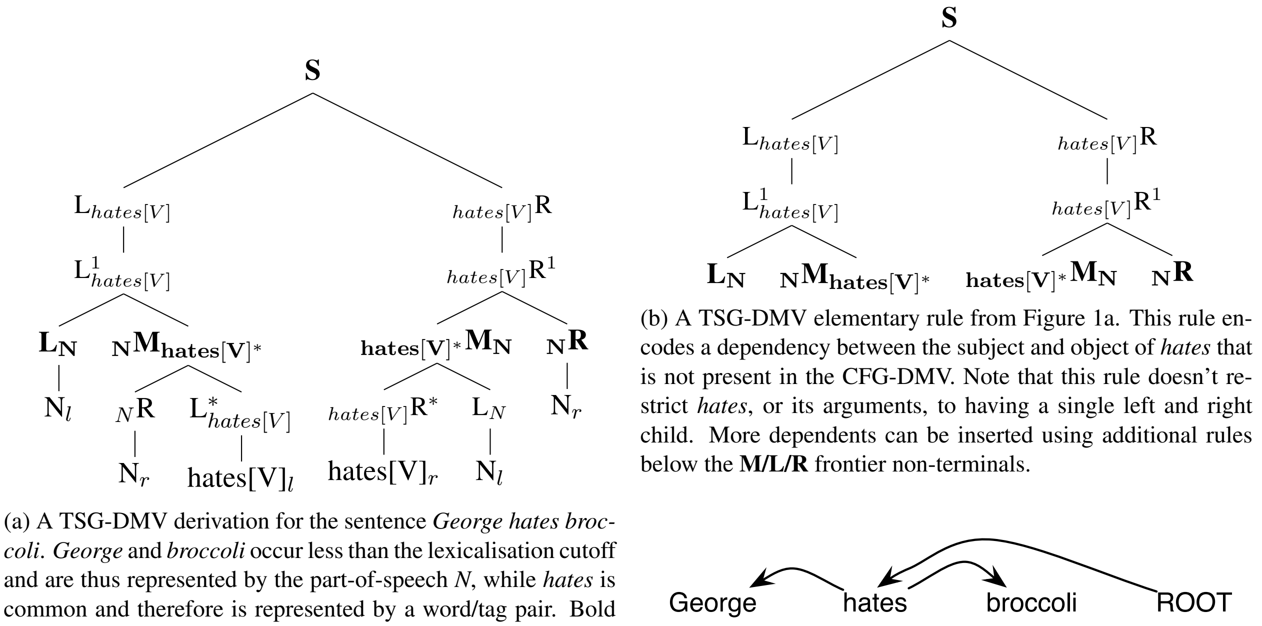
CFG Rule	DMV Distribution	Description
$\mathrm{S} \rightarrow \mathrm{L}_H\ {}_H\mathrm{R}$	$p(root = H)$	The head of the sentence is H.
$\mathrm{L}_H \rightarrow \mathrm{H}_l$	$p(STOP \mid dir = L, head = H, val = 0)$	H has no left children.
$\mathrm{L}_H \rightarrow \mathrm{L}^1_H$	$p(CONT \mid dir = L, head = H, val = 0)$	H has at least one left child.
$\mathrm{L}^*_H \rightarrow \mathrm{H}_l$	$p(STOP \mid dir = L, head = H, val = 1)$	H has no more left children.
$\mathrm{L}^*_H \rightarrow \mathrm{L}^1_H$	$p(CONT \mid dir = L, head = H, val = 1)$	H has another left child.
${}_H\mathrm{R} \rightarrow \mathrm{H}_r$	$p(STOP \mid dir = R, head = H, val = 0)$	H has no right children.
${}_H\mathrm{R} \rightarrow {}_H\mathrm{R}^1$	$p(CONT \mid dir = R, head = H, val = 0)$	H has at least one right child.
${}_H\mathrm{R}^* \rightarrow \mathrm{H}_r$	$p(STOP \mid dir = R, head = H, val = 1)$	H has no more right children.
${}_H\mathrm{R}^* \rightarrow {}_H\mathrm{R}^1$	$p(CONT \mid dir = R, head = H, val = 1)$	H has another right child.
$\mathrm{L}^1_H \rightarrow \mathrm{L}_C\ {}_C\mathrm{M}_{H^*}$	$p(C \mid dir = L, head = H)$	C is a left child of H.
${}_H\mathrm{R}^1 \rightarrow {}_{H^*}\mathrm{M}_C\ {}_C\mathrm{R}$	$p(C \mid dir = R, head = H)$	C is a right child of H.
${}_C\mathrm{M}_{H^*} \rightarrow {}_C\mathrm{R}\ \mathrm{L}^*_H$	$p = 1$	Unambiguous
${}_{H^*}\mathrm{M}_C \rightarrow {}_H\mathrm{R}^*\ \mathrm{L}_C$	$p = 1$	Unambiguous

Table 1: The CFG-DMV grammar schema. Note that the actual CFG is created by instantiating these templates with part-of-speech tags observed in the data for the variables H and C. Valency (*val*) can take the value 0 (no attachment in the direction (*dir*) d) and 1 (one or more attachment). L and R indicates child dependents left or right of the parent; superscripts encode the stopping and valency distributions, X^1 indicates that the head will continue to attach more children and X^* that it has already attached a child.

2 Background

The most successful framework for unsupervised dependency induction is the Dependency Model with Valence (DMV) (Klein and Manning, 2004). This model has been adapted and extended by a number of authors and currently represents the state-of-the-art for dependency induction (Cohen and Smith, 2009; Headden III et al., 2009). Eisner (2000) introduced the *split-head* algorithm which permits efficient $\mathcal{O}(|\mathbf{w}|^3)$ parsing complexity by replicating (splitting) each terminal and processing left and right dependents separately. We employ the related *fold-unfold* representation of Johnson (2007) that defines a CFG equivalent of the split-head parsing algorithm, allowing us to easily adapt CFG-based grammar models to dependency grammar. Table 1 shows the equivalent CFG grammar for the DMV model (CFG-DMV) using the unfold-fold transformation. The key insight to understanding the non-terminals in this grammar is that the subscripts encode the terminals at the boundaries of the span of that non-terminal. For example the non-terminal L_H encodes that the right most terminal spanned by this constituent is H (and the reverse for ${}_H\mathrm{R}$), while ${}_A\mathrm{M}_B$ encodes that A and B are the left-most and right-most terminals of the span. The * and ^{1} superscripts are used to encode the valency of the head, both indicate that the head has at least one attached dependent in the specified direction. This grammar allows $\mathcal{O}(|\mathbf{w}|^3)$ parsing complexity which follows from the terminals of the dependency tree being observed, such that each span of the parse chart uniquely specifies its possible heads (either the leftmost, rightmost or both) and therefore the number of possible non-terminals for each span is constant. The transform is illustrated in figures 1a and 1c which show the CFG tree for an example sentence and the equivalent dependency tree.

Normally DMV based models have been trained on part-of-speech tags of the words in a sentence, rather than the words themselves. Headden III et al. (2009) showed that performance could be improved by including high frequency words as well as tags in their model. In this paper we refer to such models as *lexicalised*; words which occur more than one hundred times in the training corpus are represented by a word/tag pair, while those less frequent are represented simply by their tags. We are also able to show that this basic approach to lexicalisation improves the performance of our models.



(a) A TSG-DMV derivation for the sentence *George hates broccoli*. *George* and *broccoli* occur less than the lexicalisation cutoff and are thus represented by the part-of-speech *N*, while *hates* is common and therefore is represented by a word/tag pair. Bold nodes indicate frontier nodes of elementary trees.

(b) A TSG-DMV elementary rule from Figure 1a. This rule encodes a dependency between the subject and object of *hates* that is not present in the CFG-DMV. Note that this rule doesn't restrict *hates*, or its arguments, to having a single left and right child. More dependents can be inserted using additional rules below the **M/L/R** frontier non-terminals.

(c) A traditional dependency tree representation of the parse tree in Figure 1a before applying the lexicalisation cutoff.

Figure 1: TSG-DMV representation of dependency trees.

3 Lexicalised TSG-DMV

The models we investigate in this paper build upon the CFG-DMV by defining a *Tree Substitution Grammar* (TSG) over the space of CFG rules. A TSG is a 4-tuple, $G = (T, N, S, R)$, where T is a set of *terminal symbols*, N is a set of *non-terminal symbols*, $S \in N$ is the distinguished *root non-terminal* and R is a set of productions (rules). The productions take the form of *elementary trees* – tree fragments of height ≥ 1, where each internal node is labelled with a non-terminal and each leaf is labelled with either a terminal or a non-terminal. Non-terminal leaves are called *frontier non-terminals* and form the substitution sites in the generative process of creating trees with the grammar.

A *derivation* creates a tree by starting with the root symbol and rewriting (substituting) it with an elementary tree, then continuing to rewrite frontier non-terminals with elementary trees until there are no remaining frontier non-terminals. We can represent derivations as sequences of elementary trees, $\mathbf{e}$, by specifying that during the generation of the tree each elementary tree is substituted for the left-most frontier non-terminal. Figure 1a shows a TSG derivation for the dependency tree in Figure 1c where bold nonterminal labels denote substitution sites (root/frontier nodes in the elementary trees).

The probability of a derivation, $\mathbf{e}$, is the product of the probabilities of its component rules,

$$P(\mathbf{e}) = \prod_{c \to e \in \mathbf{e}} P(e|c)\,. \qquad (1)$$

where each rewrite is assumed conditionally independent of all others given its root nonterminal, $c = \text{root}(e)$. The probability of a tree, t, and string of words, $\mathbf{w}$, are

$$P(t) = \sum_{\mathbf{e}:\text{tree}(\mathbf{e})=t} P(\mathbf{e}) \text{ and } P(\mathbf{w}) = \sum_{t:\text{yield}(t)=\mathbf{w}} P(t)\,,$$

respectively, where tree($\mathbf{e}$) returns the tree for the derivation $\mathbf{e}$ and yield(t) returns the string of terminal symbols at the leaves of t.

A *Probabilistic Tree Substitution Grammar* (PTSG), like a PCFG, assigns a probability to each rule in the grammar, denoted $P(e|c)$. The probability of a derivation, $\mathbf{e}$, is the product of the probabilities of its component rules. Estimating a PTSG requires learning the sufficient statistics for $P(e|c)$ in (1) based on a training sample. Parsing involves finding the most probable tree for a given string ($\arg\max_t P(t|\mathbf{w})$). This is typically approximated by finding the most probable derivation which can be done efficiently using the CYK algorithm.

3.1 Model

In this work we propose the Tree Substitution Grammar Dependency Model with Valence (TSG-DMV). We define a hierarchical non-parametric TSG model on the space of parse trees licensed by the CFG grammar in Table 1. Our model is a generalisation of that of Cohn et al. (2009) and Cohn et al. (2011). We extend those works by moving from a single level Dirichlet Process (DP) distribution over rules to a multi-level Pitman-Yor Process (PYP), and including lexicalisation. The PYP has been shown to generate distributions particularly well suited to modelling language (Teh, 2006; Goldwater et al., 2006). Teh (2006) used a hierarchical PYP to model backoff in language models, we leverage this same capability to model backoff in TSG rules. This effectively allows smoothing from lexicalised to unlexicalised grammars, and from TSG to CFG rules.

Here we describe our deepest model which has a four level hierarchy, depicted graphically in Table 2. In Section 5 we evaluate different subsets of this hierarchy. The topmost level of our model describes lexicalised elementary elementary fragments (e) as produced by a PYP,

$$\begin{aligned} e|c &\sim G_c \\ G_c|a_c, b_c, \mathrm{P}^{\mathrm{lcfg}} &\sim \mathrm{PYP}(a_c, b_c, \mathrm{P}^{\mathrm{lcfg}}(\cdot|c)) \,, \end{aligned}$$

where a_c and b_c control the strength of the backoff distribution $\mathrm{P}^{\mathrm{lcfg}}$. The space of lexicalised TSG rules will inevitably be very sparse, so the base distribution $\mathrm{P}^{\mathrm{lcfg}}$ backs-off to calculating the probability of a TSG rules as the product of the CFG rules it contains, multiplied by a geometric distribution over the size of the rule.

$$\begin{aligned} \mathrm{P}^{\mathrm{lcfg}}(e|c) &= \prod_{f \in \mathrm{F}(e)} s_{f_c} \prod_{i \in \mathrm{I}(e)} (1 - s_{i_c}) \\ &\quad \times A(\text{lex-cfg-rules}(e|c)) \\ \alpha|c &\sim A_c \\ A_c|a_c^{\mathrm{lcfg}}, b_c^{\mathrm{lcfg}}, \mathrm{P}^{\mathrm{cfg}} &\sim \mathrm{PYP}(a_c^{\mathrm{lcfg}}, b_c^{\mathrm{lcfg}}, \mathrm{P}^{\mathrm{cfg}}(\cdot|c)), \end{aligned}$$

where $I(e)$ are the set of internal nodes in e excluding the root, $F(e)$ are the set of frontier non-terminal nodes, and c_i is the non-terminal symbol for node i and s_c is the probability of stopping expanding a node labelled c. The function lex-cfg-rules$(e|c)$ returns the CFG rules internal to e, each of the form $c' \rightarrow \alpha$; each CFG rule is drawn from the back-off distribution, $A_{c'}$. We treat s_c as a parameter which is estimated during training, as described in Section 4.2.

The next level of backoff ($\mathrm{P}^{\mathrm{cfg}}$) removes the lexicalisation from the CFG rules, describing the generation of a lexicalised rule by first generating an unlexicalised rule from a PYP, then generating the lexicalisaton from a uniform distribution over words:[1]

$$\begin{aligned} \mathrm{P}^{\mathrm{cfg}}(\alpha|c) &= B(\text{unlex}(\alpha)|\text{unlex}(c)) \\ &\quad \times \frac{1}{|\mathbf{w}|^{|\alpha|}} \\ \alpha'|c' &\sim B_{c'} \\ B_{c'}|a_{c'}^{\mathrm{cfg}}, b_{c'}^{\mathrm{cfg}}, \mathrm{P}^{\mathrm{sh}} &\sim \mathrm{PYP}(a_{c'}^{\mathrm{cfg}}, b_{c'}^{\mathrm{cfg}}, \mathrm{P}^{\mathrm{sh}}(\cdot|c')), \end{aligned}$$

where unlex$(\cdot)$ removes the lexicalisation from non-terminals leaving only the tags.

The final base distribution over CFG-DMV rules (P^{sh}) is inspired by the *skip-head* smoothing model of Headden III et al. (2009). This model showed that smoothing the DMV by removing the heads from the CFG rules significantly improved performance. We replicate this behavior through a final level in our hierarchy which generates the CFG rules without their heads, then generates the heads from a uniform distribution:

$$\begin{aligned} \mathrm{P}^{\mathrm{sh}}(\alpha|c) &= C(\text{drop-head}(c \rightarrow \alpha)) \times \frac{1}{|P|} \\ \alpha|c &\sim C_c \\ C_c|a_c^{\mathrm{sh}}, b_c^{\mathrm{sh}} &\sim \mathrm{PYP}(a_c^{\mathrm{sh}}, b_c^{\mathrm{sh}}, \text{Uniform}(\cdot|c)), \end{aligned}$$

where drop-head$(\cdot)$ removes the symbols that mark the head on the CFG rules, and P is the set of part-of-speech tags. Each stage of backoff is illustrated in Table 2, showing the rules generated from the TSG elementary tree in Figure 1b.

Note that while the supervised model of Cohn et al. (2009) used a fixed back-off PCFG distribution, this model implicitly infers this distribution within

[1] All unlexicalised words are actually given the generic UNK symbol as their lexicalisation.

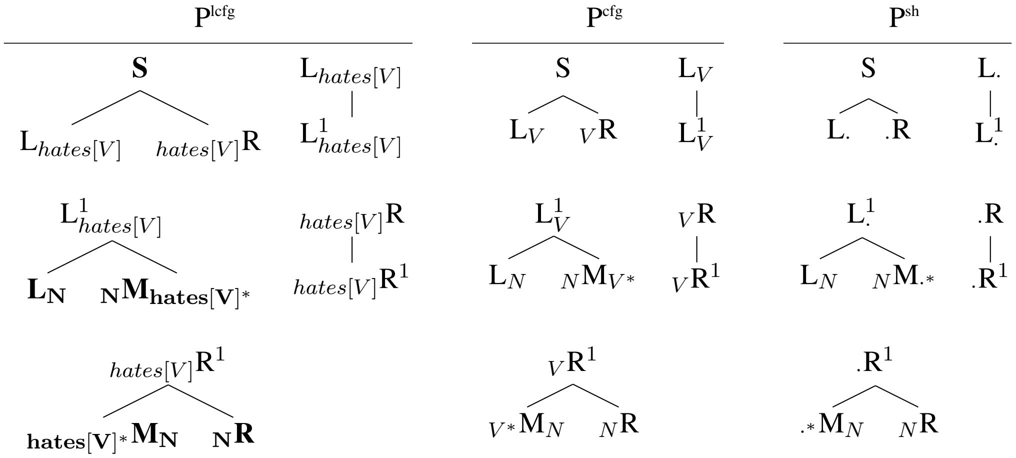


Table 2: Backoff trees for the elementary tree in Figure 1b.

its hierarchy, essentially learning the DMV model embedded in the TSG.

In this application to dependency grammar our model is capable of learning tree fragments which group CFG parameters. As such the model can learn to condition dependency links on the valence, e.g. by combining $\mathrm{L}_H \rightarrow \mathrm{L}^1_H$ and $\mathrm{L}^1_H \rightarrow \mathrm{L}_C\ {}_C\mathrm{M}_{H^*}$ rules into a single fragment the model can learn a parameter that the leftmost child of H is C. By linking together multiple L^1_H or ${}_H\mathrm{R}^1$ non-terminals the model can learn groups of dependencies that occur together, e.g. tree fragments representing the complete preferred argument frame of a verb.

4 Inference

4.1 Training

To train our model we use Markov Chain Monte Carlo sampling (Geman and Geman, 1984). Where previous supervised TSG models (Cohn et al., 2009) permit an efficient local sampler, the lack of an observed parse tree in our unsupervised model makes this sampler not applicable. Instead we use a recently proposed blocked Metroplis-Hastings (MH) sampler (Cohn and Blunsom, 2010) which exploits a factorisation of the derivation probabilities such that whole trees can be sampled efficiently. See Cohn and Blunsom (2010) for details. That algorithm is applied using a dynamic program over an observed tree, the generalisation to our situation of an inside pass over the space of all trees is straightforward.

A final consideration is the initialisation of the sampler. Klein and Manning (2004) emphasised the importance of the initialiser for achieving good performance with their model. We employ the same *harmonic* initialiser as described in that work. The initial derivations for our sampler are the Viterbi derivations under the CFG parameterised according to this initialiser.

4.2 Sampling hyperparameters

We treat the hyper-parameters $\{(a^x_c, b^x_c, s_c), c \in N\}$ as random variables in our model and infer their values during training. We choose quite vague priors for each hyper-parameter, encoding our lack of information about their values.

We place prior distributions on the PYP discount a_c and concentration b_c hyperparamters and sample their values using a slice sampler. We use the range doubling slice sampling technique of (Neal, 2003) to draw a new sample of a'_c from its conditional distribution.[2] For the discount parameters a_c we employ a uniform Beta distribution, as we have no strong prior knowledge of what its value should be ($a_c \sim \mathrm{Beta}(1, 1)$). Similarly, we treat the concentration parameters, b_c, as being generated by a vague gamma prior, $b_c \sim \mathrm{Gamma}(1, 1)$, and sample a new value b'_c using the same slice-sampling approach as for a_c:

$$P(b_c|\mathbf{z}) \propto P(\mathbf{z}|b_c) \times \mathrm{Gamma}(b_c|1, 1).$$

[2] We made use of the slice sampler included in Mark Johnson's Adaptor Grammar implementation http://www.cog.brown.edu/~mj/Software.htm.

Corpus	Words	Sentences
Sections 2-21 ($\|\mathbf{x}\| \leq 10$)	42505	6007
Section 22 ($\|\mathbf{x}\| \leq 10$)	1805	258
Section 23 ($\|\mathbf{x}\| \leq 10$)	2649	398
Section 23 ($\|\mathbf{x}\| \leq \infty$)	49368	2416

Table 3: Corpus statistics for the training and testing data for the TSG-DMV model. All models are trained on the gold standard part-of-speech tags after removing punctuation.

We use a vague Beta prior for the stopping probabilities in $\mathrm{P}^{\mathrm{lcfg}}$, $s_c \sim \mathrm{Beta}(1, 1)$.

All the hyper-parameters are resampled after every 10th sample of the corpus derivations.

4.3 Parsing

Unfortunately finding the maximising parse tree for a string under our TSG-DMV model is intractable due to the inter-rule dependencies created by the PYP formulation. Previous work has used Monte Carlo techniques to sample for one of the maximum probability parse (MPP), maximum probability derivation (MPD) or maximum marginal parse (MMP) (Cohn et al., 2009; Bod, 2006). We take a simpler approach and use the Viterbi algorithm to calculate the MPD under an approximating TSG defined by the last set of derivations sampled for the corpus during training. Our results indicate that this is a reasonable approximation, though the experience of other researchers suggests that calculating the MMP under the approximating TSG may also be beneficial for DMV (Cohen et al., 2008).

5 Experiments

We follow the standard evaluation regime for DMV style models by performing experiments on the text of the WSJ section of the Penn. Treebank (Marcus et al., 1993) and reporting head attachment accuracy. Like previous work we pre-process the training and test data to remove punctuation, training our unlexicalised models on the gold-standard part-of-speech tags, and including words occurring more than 100 times in our lexicalised models (Headden III et al., 2009). It is very difficult for an unsupervised model to learn from long training sentences as they contain a great deal of ambiguity, therefore the majority of DMV based models have been trained on sentences restricted in length to ≤ 10 tokens.[3] This has the added benefit of decreasing the runtime for experiments. We present experiments with this training scenario. The training data comes from sections 2-21, while section 23 is used for evaluation. An advantage of our sampling based approach over previous work is that we infer all the hyperparameters, as such we don't require the use of section 22 for tuning the model.

The models are evaluated in terms of head attachment accuracy (the percentage of correctly predicted head indexes for each token in the test data), on two subsets of the testing data. Although we can argue that unsupervised models are better learnt from short sentences, it is much harder to argue that we don't then need to be able to parse long sentences with a trained model. The most commonly employed test set mirrors the training data by only including sentences ≤ 10. In this work we focus on the accuracy of our models on the whole of section 23, without any pruning for length. The training and testing corpora statistics are presented in Table 3. Subsequent to the evaluation reported in Table 4 we use section 22 to report the correlation between heldout accuracy and the model log-likelihood (LLH) for analytic purposes.

As we are using a sampler during training, the result of any single run is non-deterministic and will exhibit a degree of variance. All our reported results are the mean and standard deviation (σ) from forty sampling runs.

5.1 Discussion

Table 4 shows the head attachment accuracy results for our TSG-DMV, plus many other significant previously proposed models. The subset of hierarchical priors used by each model is noted in brackets.

The performance of our models is extremely encouraging, particularly the fact that it achieves the highest reported accuracy on the full test set by a considerable margin. On the $|\mathbf{w}| \leq 10$ test set all the TSG-DMVs are second only to the L-EVG model of Headden III et al. (2009). The L-EVG model extends DMV by adding additional lexicalisation,

[3] See Spitkovsky et al. (2010a) for an exception to this rule.

	Directed Attachment Accuracy on WSJ23	
Model	$\lvert\mathbf{w}\rvert \leq 10$	$\lvert\mathbf{w}\rvert \leq \infty$
Attach-Right	38.4	31.7
EM (Klein and Manning, 2004)	46.1	35.9
Dirichlet (Cohen et al., 2008)	46.1	36.9
LN (Cohen et al., 2008)	59.4	40.5
SLN, TIE V&N (Cohen and Smith, 2009)	61.3	41.4
DMV (Headden III et al., 2009)	$55.7_{\sigma=8.0}$	-
DMV smoothed (Headden III et al., 2009)	$61.2_{\sigma=1.2}$	-
EVG smoothed (Headden III et al., 2009)	$65.0_{\sigma=5.7}$	-
L-EVG smoothed (Headden III et al., 2009)	$\mathbf{68.8}_{\sigma=4.5}$	-
Less is More (Spitkovsky et al., 2010a)	56.2	44.1
Leap Frog (Spitkovsky et al., 2010a)	57.1	45.0
Viterbi EM (Spitkovsky et al., 2010b)	65.3	47.9
Hypertext Markup (Spitkovsky et al., 2010c)	69.3	50.4
Adaptor Grammar (Cohen et al., 2010)	50.2	-
TSG-DMV ($\mathrm{P}^{\mathrm{cfg}}$)	$65.9_{\sigma=2.4}$	$53.1_{\sigma=2.4}$
TSG-DMV ($\mathrm{P}^{\mathrm{cfg}}$, P^{sh})	$65.1_{\sigma=2.2}$	$51.5_{\sigma=2.0}$
LexTSG-DMV ($\mathrm{P}^{\mathrm{lcfg}}$, $\mathrm{P}^{\mathrm{cfg}}$)	$67.2_{\sigma=1.4}$	$55.2_{\sigma=2.2}$
LexTSG-DMV ($\mathrm{P}^{\mathrm{lcfg}}$, $\mathrm{P}^{\mathrm{cfg}}$, P^{sh})	$67.7_{\sigma=1.5}$	$\mathbf{55.7}_{\sigma=2.0}$
Supervised MLE (Cohen and Smith, 2009)	84.5	68.8

Table 4: Mean and variance for the head attachment accuracy of our TSG-DMV models (highlighted) with varying backoff paths, and many other high performing models. Citations indicate where the model and result were reported. Our models labelled TSG used an unlexicalised top level G_c PYP, while those labelled LexTSG used the full lexicalised G_c.

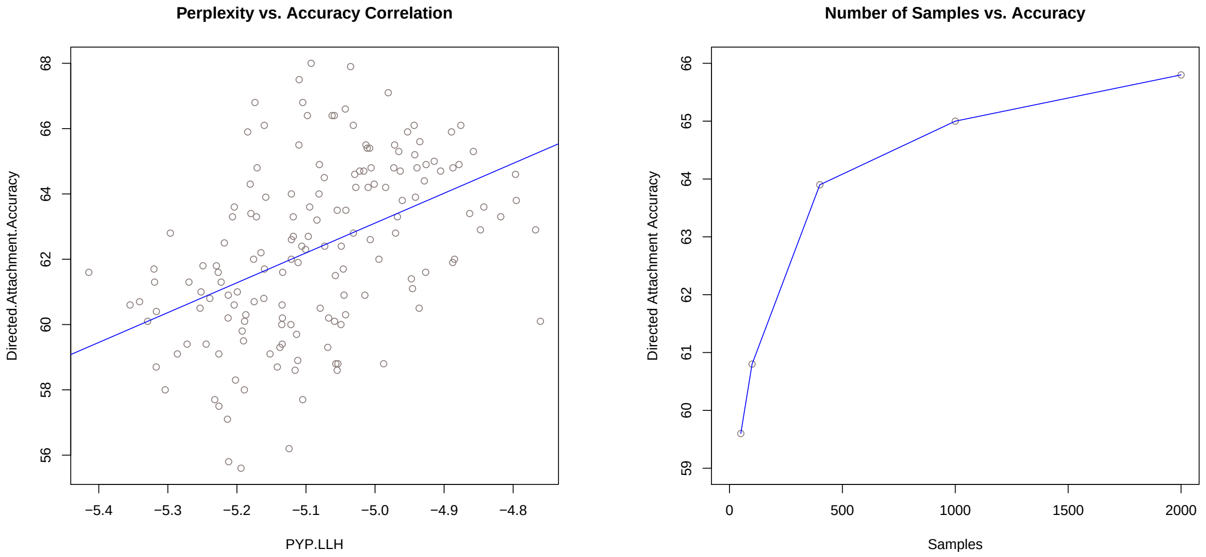
valency conditioning, interpolated back-off smoothing and a random initialiser. In particular Headden III et al. (2009) shows that the random initialiser is crucial for good performance, however this initialiser requires training 1000 models to select a single best model for evaluation and results in considerable variance in test set performance. Note also that our model exhibits considerably less variance than those induced using this random initialiser, suggesting that the combination of the harmonic initialiser and blocked-MH sampling may be a more practicable training regime.

The recently proposed Adaptor Grammar DMV model of Cohen et al. (2010) is similar in many way to our TSG model, incorporating a Pitman Yor prior over units larger than CFG rules. As such it is surprising that our model is performing significantly better than this model. We can identify a number of differences that may impact these results: the Adaptor Grammar model is trained using variational inference with the space of tree fragments truncated, while we employ a sampler which can nominally explore the full space of tree fragments; and the adapted tree fragments must be complete subtrees (i.e. they don't contain variables), whereas our model can make use of arbitrary tree fragments. An interesting avenue for further research would be to extend the variational algorithm of Cohen et al. (2010) to our TSG model, possibly speeding inference and allowing easier parallelisation.

In Figure 2a we graph the model LLH on the training data versus the head attachment accuracy on the heldout set. The graph was generated by running 160 models for varying numbers of samples and evaluating their accuracy. This graph indicates that the improvements in the posterior probability of the model are correlated with the evaluation, though the correlation is not as high as we might require in order to use LLH as a model selection criteria similar to Headden III et al. (2009). Further refinements to the model could improve this correlation.

The scaling perfomance of the model as the number of samples is increased is shown in Figure 2b. Performance improves as the training data is sampled for longer, and continues to trend upwards beyond 1000 samples (the point for which we've reported results in Table 4). This suggests that longer sampling runs – and better inference techniques – could yield further improvements.

For further analysis Table 5 shows the accuracy of the model at predicting the head for frequent types, while Table 6 shows the performance on dependencies of various lengths. We emphasise that these results are for the single best performing sampler run on the heldout corpus and there is considerable variation in the analyses produced by each sampler. Unsurprisingly, the model appears to be more accurate when predicting short dependencies, a result that is also reflected in the per type accuracies. The model is relatively good at identifying the root verb in each sentence, especially those headed by past tense verbs (VBD, *was*), and to a lesser degree VBPs (*are*). Conjunctions such as *and* pose a particular difficulty when evaluating dependency models as the correct modelling of these remains a



(a) Correlation ($R^2 = 0.2$) between the training LLH of the PYP Model and heldout directed head attachment accuracy (WSJ Section 22, $|\mathbf{w}| \leq 10$) for LexTSG-DMV ($\mathrm{P}^{\mathrm{lcfg}}$, $\mathrm{P}^{\mathrm{cfg}}$, P^{sh}).

(b) Mean heldout directed head attachment accuracy (WSJ Section 22, $|\mathbf{w}| \leq 10$) versus the number of samples used during training for LexTSG-DMV ($\mathrm{P}^{\mathrm{lcfg}}$, $\mathrm{P}^{\mathrm{cfg}}$, P^{sh}).

Figure 2

contentious linguistic issue and it's not clear what the 'correct' analysis should be. Our model gets a respectable 75% accuracy for *and* conjunctions, but for conjunctions (CC) as a whole, the model performs poorly (39%).

Table 7 list the most frequent TSG rules lexicalised with *has*. The most frequent rule is simply the single level equivalent of the DMV terminal rule for *has*. Almost as frequent is rule 3, here the grammar incorporates the terminal into a larger elementary fragment, encoding that it is the head of the past participle occuring immediately to it's right. This shows the model's ability to learn the verb's argument position conditioned on both the head and child type, something lacking in DMV. Rule 7 further refines this preferred analysis for *has been* by lexicalising both the head and child. Rules (4,5,8,10) employ similar conditioning for proper and ordinary nouns heading noun phrases to the left of *has*. We believe that it is the ability of the TSG to encode stronger constraints on argument positions that leads to the model's higher accuracy on longer sentences, while other models do well on shorter sentences but relatively poorly on longer ones (Spitkovsky et al., 2010c).

6 Conclusion

In this paper we have made two significant contributions to probabilistic modelling and grammar induction. We have shown that it is possible to successfully learn hierarchical Pitman-Yor models that encode deep and complex backoff paths over highly structured latent spaces. By applying these models to the induction of dependency grammars we have also been able to advance the state-of-the-art, increasing the head attachment accuracy on section 23 of the Wall Street Journal Corpus by more than 5%.

Further gains in performance may come from an exploration of the backoff paths employed within the model. In particular more extensive experimentation with alternate priors and larger training data may allow the removal of the lexicalisation cutoff which is currently in place to counter sparsity.

We envisage that in future many grammar formalisms that have been shown to be effective in supervised parsing, such as categorial, unification and tree adjoining grammars, will prove amenable to unsupervised induction using the hierarchical nonparametric modelling approaches we have demonstrated in this paper.

	Count	LexTSG-DMV Rules		
1	94	${\mathrm{L}^*}_{has-VBZ}$	$\rightarrow$	$({\mathrm{L}^*}_{has-VBZ}\ \text{has-VBZ}_l)$
2	74	$\mathrm{L}^1_{has-VBZ}$	$\rightarrow$	$(\mathrm{L}^1_{has-VBZ}\ (\mathrm{L}_{NN}\ \mathrm{L}^1_{NN})\ {}_{NN}\mathrm{M}_{has-VBZ^*})$
3	71	${}_{has-VBZ^*}\mathrm{M}_{VBN}$	$\rightarrow$	$({}_{has-VBZ^*}\mathrm{M}_{VBN}\ ({}_{has-VBZ}\mathrm{R}^*\ \text{has-VBZ}_r)\ \mathrm{L}_{VBN})$
4	54	${}_{NN}\mathrm{M}_{has-VBZ^*}$	$\rightarrow$	$({}_{NN}\mathrm{M}_{has-VBZ^*}\ {}_{NN}\mathrm{R}\ ({\mathrm{L}^*}_{has-VBZ}\ \text{has-VBZ}_l))$
5	36	${}_{NN}\mathrm{M}_{has-VBZ^*}$	$\rightarrow$	$({}_{NN}\mathrm{M}_{has-VBZ^*}\ {}_{NN}\mathrm{R}\ {\mathrm{L}^*}_{has-VBZ})$
6	36	${}_{has-VBZ}\mathrm{R}^*$	$\rightarrow$	$({}_{has-VBZ}\mathrm{R}^*\ ({}_{has-VBZ}\mathrm{R}^1\ {}_{has-VBZ^*}\mathrm{M}_{VBN}\ ({}_{VBN}\mathrm{R}\ \text{VBN}_r)))$
7	30	${}_{has-VBZ^*}\mathrm{M}_{been-VBN}$	$\rightarrow$	$({}_{has-VBZ^*}\mathrm{M}_{been-VBN}\ ({}_{has-VBZ}\mathrm{R}^*\ \text{has-VBZ}_r)\ \mathrm{L}_{been-VBN})$
8	27	${}_{NNP}\mathrm{M}_{has-VBZ^*}$	$\rightarrow$	$({}_{NNP}\mathrm{M}_{has-VBZ^*}\ {}_{NNP}\mathrm{R}\ ({\mathrm{L}^*}_{has-VBZ}\ \text{has-VBZ}_l))$
9	25	${}_{has-VBZ}\mathrm{R}$	$\rightarrow$	$({}_{has-VBZ}\mathrm{R}\ ({}_{has-VBZ}\mathrm{R}^1\ {}_{has-VBZ^*}\mathrm{M}_{NNS}\ ({}_{NNS}\mathrm{R}\ {}_{NNS}\mathrm{R}^1)))$
10	18	$\mathrm{L}^1_{has-VBZ}$	$\rightarrow$	$(\mathrm{L}^1_{has-VBZ}\ \mathrm{L}_{NNP}\ {}_{NNP}\mathrm{M}_{has-VBZ^*})$

Table 7: The ten most frequent LexTSG-DMV rules in a final training sample that contain *has*.

References

Rens Bod. 2006. An all-subtrees approach to unsupervised parsing. In *Proc. of the 44th Annual Meeting of the ACL and 21st International Conference on Computational Linguistics (COLING/ACL-2006)*, pages 865–872, Sydney, Australia, July.

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.

Alexander Clark. 2001. Unsupervised induction of stochastic context-free grammars using distributional clustering. In *ConLL '01: Proceedings of the 2001 workshop on Computational Natural Language Learning*, pages 1–8. Association for Computational Linguistics.

Shay B. Cohen and Noah A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *NAACL '09: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 74–82, Morristown, NJ, USA. Association for Computational Linguistics.

Shay B. Cohen, Kevin Gimpel, and Noah A. Smith. 2008. Logistic normal priors for unsupervised probabilistic grammar induction. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Lon Bottou, editors, *NIPS*, pages 321–328. MIT Press.

Shay B. Cohen, David M. Blei, and Noah A. Smith. 2010. Variational inference for adaptor grammars. In *Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Trevor Cohn and Phil Blunsom. 2010. Blocked inference in Bayesian tree substitution grammars. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, page To Appear, Uppsala, Sweden.

Trevor Cohn, Sharon Goldwater, and Phil Blunsom. 2009. Inducing compact but accurate tree-substitution grammars. In *NAACL '09: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics on ZZZ*, pages 548–556, Morristown, NJ, USA. Association for Computational Linguistics.

Trevor Cohn, Phil Blunsom, and Sharon Goldwater. 2011. Inducing tree-substitution grammars. *Journal of Machine Learning Research*. To Appear.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In Harry Bunt and Anton Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62. Kluwer Academic Publishers, October.

Stuart Geman and Donald Geman. 1984. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.

Sharon Goldwater, Tom Griffiths, and Mark Johnson. 2006. 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. MIT Press, Cambridge, MA.

William P. Headden III, Mark Johnson, and David McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 101–109, Boulder, Colorado, June.

Child Tag	Predicted Head Correct	Accuracy (%)
NN	181	0.64
NNP	130	0.71
DT	127	0.87
NNS	108	0.72
VBD	108	0.81
JJ	106	0.80
IN	81	0.55
RB	65	0.61
PRP	64	0.97
VBZ	47	0.80
VBN	36	0.86
VBP	30	0.77
CD	26	0.23
VB	25	0.68
the	42	0.88
was	29	0.97
The	25	0.83
of	18	0.78
a	18	0.90
to	17	0.50
in	16	0.89
is	15	0.79
n't	15	0.83
were	12	0.86
are	11	0.92
It	11	1.00
for	9	0.64
and	9	0.75
's	9	1.00

Table 5: Per tag type predicted count and accuracy, for the most frequent 15 un/lexicalised tokens on the WSJ Section 22 $|\mathbf{w}| \leq 10$ heldout set (LexTSG-DMV $(P^{lcfg}, P^{cfg}, P^{sh})$).

Mark Johnson. 2007. Transforming projective bilexical dependency grammars into efficiently-parsable CFGs with unfold-fold. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 168–175, Prague, Czech Republic, June. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: models of dependency and constituency. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 478.

Distance	Precision	Recall	F1
1	0.70	0.75	0.72
2	0.70	0.62	0.65
3	0.66	0.62	0.64
4	0.56	0.56	0.56
5	0.53	0.49	0.51
6	0.59	0.66	0.62
7	0.50	0.44	0.47
8	0.57	0.33	0.42
9	0.67	0.40	0.50
10	1.00	0.17	0.29

Table 6: Link distance precision, recall and f-score, on the WSJ Section 22 $|\mathbf{w}| \leq 10$ heldout set.

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

Ryan McDonald. 2006. *Discriminative Training and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvania.

Igor′ A. Mel′čuk. 1988. *Dependency Syntax: theory and practice*. State University of New York Press, Albany.

Radford Neal. 2003. Slice sampling. *Annals of Statistics*, 31:705–767.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010a. From Baby Steps to Leapfrog: How "Less is More" in unsupervised dependency parsing. In *Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Valentin I. Spitkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christopher D. Manning. 2010b. Viterbi training improves unsupervised dependency parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning (CoNLL-2010)*.

Valentin I. Spitkovsky, Daniel Jurafsky, and Hiyan Alshawi. 2010c. Profiting from mark-up: Hyper-text annotations for guided parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*.

Y. W. Teh. 2006. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 985–992.

It Depends on the Translation: Unsupervised Dependency Parsing via Word Alignment

Samuel Brody
Dept. of Biomedical Informatics
Columbia University
samuel.brody@dbmi.columbia.edu

Abstract

We reveal a previously unnoticed connection between dependency parsing and statistical machine translation (SMT), by formulating the dependency parsing task as a problem of word alignment. Furthermore, we show that two well known models for these respective tasks (DMV and the IBM models) share common modeling assumptions. This motivates us to develop an alignment-based framework for unsupervised dependency parsing. The framework (which will be made publicly available) is flexible, modular and easy to extend. Using this framework, we implement several algorithms based on the IBM alignment models, which prove surprisingly effective on the dependency parsing task, and demonstrate the potential of the alignment-based approach.

1 Introduction

Both statistical machine translation (SMT) and unsupervised dependency parsing have seen a surge of interest in recent years, as the need for large scale data processing has increased. The problems addressed by each of the fields seem quite different at first glance. However, in this paper, we reveal a strong connection between them and show that the problem of dependency parsing can be formulated as one of word alignment within the sentence. Furthermore, we show that the two models that are arguably the most influential in their respective fields, the IBM models 1-3 (Brown et al., 1993) and Klein and Manning's (2004) *Dependency Model with Valence* (DMV), share a common set of modeling assumptions.

Based on this connection, we develop a framework which uses an alignment-based approach for unsupervised dependency parsing. The framework is flexible and modular, and allows us to explore different modeling assumptions. We demonstrate these properties and the merit of the alignment-based parsing approach by implementing several dependency parsing algorithms based on the IBM alignment models and evaluating their performance on the task. Although the algorithms are not competitive with state-of-the-art systems, they outperform the right-branching baseline and approach the performance of DMV. This is especially surprising when we consider that the IBM models were not originally designed for the task. These results are encouraging and indicate that the alignment-based approach could serve as the basis for competitive dependency parsing systems, much as DMV did.

This paper offers two main contributions. First, by revealing the connection between the two tasks, we introduce a new approach to dependency parsing, and open the way for use of SMT alignment resources and tools for parsing. Our experiments with the IBM models demonstrate the potential of this approach and provide a strong motivation for further development. The second contribution is a publicly-available framework for exploring new alignment models. The framework uses Gibbs sampling techniques and includes our sampling-based implementations of the IBM models (see Section 3.4). The sampling approach makes it easy to modify the existing models and add new ones. The framework can be used both for dependency parsing and for bilingual word alignment.

The rest of the paper is structured as follows. In Section 2 we present a brief overview of those works in the fields of dependency parsing and alignment for statistical machine translation which are directly

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1214–1222,
MIT, Massachusetts, USA, 9-11 October 2010.

relevant to this paper. Section 3 describes the connection between the two problems, examines the shared assumptions of the DMV and IBM models, and describes our framework and algorithms. In Section 4 we present our experiments and discuss the results. We conclude in Section 5.

2 Background and Related Work

2.1 Unsupervised Dependency Parsing

In recent years, the field of supervised parsing has advanced tremendously, to the point where highly accurate parsers are available for many languages. However, supervised methods require the manual annotation of training data with parse trees, a process which is expensive and time consuming. Therefore, for domains and languages with minimal resources, unsupervised parsing is of great importance.

Early work in the field focused on models that made use primarily of the co-occurrence information of the head and its argument (Yuret, 1998; Paskin, 2001). The introduction of DMV by Klein and Manning (2004) represented a shift in the direction of research in the field. DMV is based on a linguistically motivated generative model, which follows common practice in *supervised* parsing and takes into consideration the distance between head and argument, as well as the valence (the capacity of a head word to attach arguments). Klein and Manning (2004) also shifted from a lexical representation of the sentences to representing them as part-of-speech sequences. DMV strongly outperformed previous models and was the first unsupervised dependency induction system to achieve accuracy above the right-branching baseline. Much subsequent work in the field has focused on modifications and extensions of DMV, and it is the basis for today's state-of-the-art systems (Cohen and Smith, 2009; Headden III et al., 2009).

2.2 Alignment for SMT

SMT treats translation as a machine learning problem. It attempts to learn a translation model from a parallel corpus composed of sentences and their translations. The IBM models (Brown et al., 1993) represent the first generation of word-based SMT models, and serve as a starting point for most current SMT systems (e.g., Moses, Koehn et al. 2007; Hiero, Chiang 2005). The models employ the notion of *alignment* between individual words in the source and translation. An example of such an alignment is given in Figure 1.

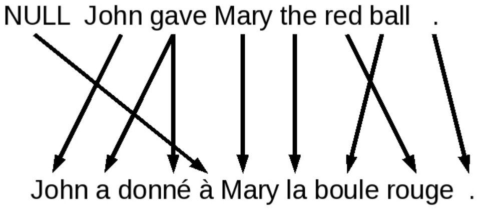

NULL John gave Mary the red ball .
John a donné à Mary la boule rouge .

Figure 1: An example of an alignment between an English sentence (top) and its French translation (bottom).

The IBM models all seek to maximize $Pr(f|e)$, the probability of a French translation f of an English sentence e. This probability is broken down by taking into account all possible alignments a between e and f, and their probabilities:

$$Pr(f|e) = \sum_{a} Pr(f,a|e) \tag{1}$$

Each of the IBM models is based on the previous one in the series, and adds another level of latent parameters which take into account a specific characteristic of the data.

3 Alignment-based Dependency Parsing

3.1 The Connection

The task of dependency parsing requires finding a parse tree for a sentence, where two words are connected by an edge if they participate in a syntactic dependency relation. When dealing with unlabeled dependencies, the exact nature of the relationship is not determined. An example of a dependency parse of a sentence is given in Figure 2 (left).

Another possible formulation of the problem is as follows. Find a set of pairwise relations (s_i, s_j) connecting a dependent word s_j with its head word s_i in the sentence. This alternate formulation allows us to view the problem as one of alignment of a sentence to itself, as shown in Figure 2 (right).

Given this perspective on the problem, it makes sense to examine existing alignment models, compare them to dependency parsing models, and see if they can be successfully employed for the dependency parsing task.

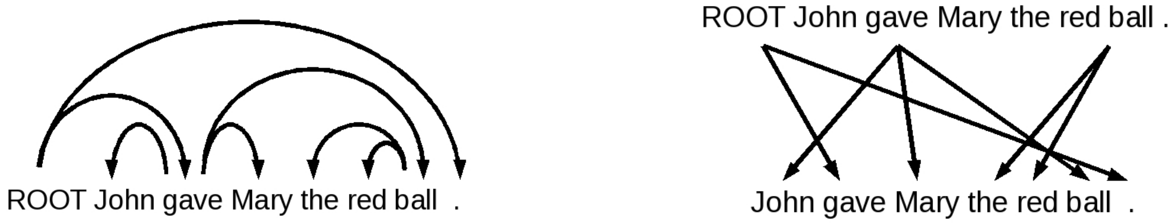

ROOT John gave Mary the red ball .
ROOT John gave Mary the red ball .
John gave Mary the red ball .

Figure 2: **Left:** An example of an unlabeled dependency parse of a sentence. **Right:** The same parse, in the form of an alignment between a head words (top) and their dependents (bottom).

3.2 Comparing IBM & DMV Assumptions

Lexical Association The core assumption of IBM Model 1 is that the lexical identities of the English and French words help determine whether they should be aligned. The same assumption is made in all the dependency models mentioned in Section 2 regarding a head and its dependent (although DMV uses word classes instead of the actual words).

Location IBM Model 2 adds the consideration of difference in location between the English and French words when considering the likelihood of alignment. One of the improvements contributing to the success of DMV was the notion of distance, which was absent from previous models (see Section 3 in Klein and Manning 2004).

Fertility IBM Model 3 adds the notion of *fertility*, or the idea that different words in the source language tend to generate different numbers of words in the target language. This corresponds to the notion of *valence*, used by Klein and Manning (2004), and the other major contributor to the success of DMV (ibid.).

Null Source The IBM models all make use of an additional “null” word in every sentence, which has special status. It is attached to words in the translation that do not correspond to a word in the source. It is treated separately when calculating distance (since it has no location) and fertility. In these characteristics, it is very similar to the “root” node, which is artificially added to parse trees and used to represent the head of words which are not dependents of any other word in the sentence.

In examining the core assumptions of the IBM models, we note that there is a strong resemblance to those of DMV. The similarity is at an abstract level since the nature of the relationship that each model attempts to detect is quite different. The IBM models look for an equivalence relationship between lexical items in two languages, whereas DMV addresses functional relationships between two elements with distinct meanings. However, both attempt to model a similar set of factors, which they posit will be important to their respective tasks[1]. This similarity motivates the work presented in the rest of the paper, i.e, exploring the use of the IBM alignment models for dependency parsing. It is important to note that the IBM models do not address many important factors relevant to the parsing task. For instance, they have no notion of a parse tree, a deficit which may lead to degenerate solutions and malformed parses. However, they serve as a good starting point for exploring the alignment approach to parsing, as well as discovering additional factors that need to be addressed under this approach.

3.3 Experimental Framework

We developed a Gibbs sampling framework for alignment-based dependency parsing[2]. The traditional approach to alignment uses Expectation Maximization (EM) to find the optimal values for the latent variables. In each iteration, it considers all possible alignments for each pair of sentences, and

[1] These abstract notions (lexical association, proximity, tendencies towards few or many relations, and allowing for unassociated items) play an important role in many relation-detection tasks (e.g., co-reference resolution, Haghighi and Klein 2010).

[2] Available for download at:
`http://people.dbmi.columbia.edu/~sab7012`

chooses the optimal one based on the current parameter estimates. The sampling method, on the other hand, only considers a small change in each step - that of re-aligning a previously aligned target word to a new source. The reason for our choice is the ease of modification of such sampling models. They allow for easy introduction of further parameters and more complex probabilistic functions, as well as Bayesian priors, all of which are likely to be helpful in development[3].

Under the sampling framework, the model provides the probability of changing the alignment $A[i]$ of a target word i from a previously aligned source word j to a new one $\hat{j}$. In all the models we consider, this probability is proportional to the ratio between the scores of the old sentence alignment A and the new one $\hat{A}$, which differs from the old only in the realignment of i to $\hat{j}$.

$$P(A[i] = j \Rightarrow A[i] = \hat{j}\,) \sim \frac{P_{model}(\hat{A})}{P_{model}(A)} \tag{2}$$

As a starting point for our dependency parsing model, we re-implemented the first three IBM models [4] in the sampling framework.

3.4 Reformulating the IBM models

IBM Model 1 According to this model, the probability of an alignment between target word i and source word $\hat{j}$ depends only on the lexical identities of the two words w_i and $w_{\hat{j}}$ respectively. This gives us equation 3.

$$\begin{aligned} P(A[i] \Rightarrow \hat{j}\,) &\sim \frac{P_{model}(\hat{A})}{P_{model}(A)} = \frac{\prod_k P(w_k, w_{A[k]})}{\prod_{k'} P(w_{k'}, w_{\hat{A}[k']})} \\ P(A[i] \Rightarrow \hat{j}\,) &\sim \frac{P(w_i, w_{\hat{j}})}{P(w_i, w_j)} \end{aligned} \tag{3}$$

In our implementation we assume the alignment follows a Chinese Restaurant Process (CRP), where the probability of w_i aligning to w_j is proportional to the number of times they have been aligned in the past (the rest of the data), as follows:

$$P(w_i, w_{\hat{j}}) = \frac{\#(w_i, w_j) + \alpha_1/V}{\#(*, w_j) + \alpha_1} \tag{4}$$

Here, $\#(w_i, w_j)$ represents the number of times the target word w_i was observed to be aligned to w_j in the rest of the data, and $*$ stands for any word, V is the size of the vocabulary, and α_1 is a hyperparameter of the CRP, which can also be viewed as a smoothing factor.

IBM Model 2 The original IBM model 2 is a distortion model that assumes that the probability of an alignment between target word i and source word $\hat{j}$ depends only on the locations of the words, i.e., the values i and $\hat{j}$, taking into account the different lengths l and m of the source and target sentences, respectively. For dependency parsing, where we align sentences to themselves, $l = m$. This gives us equation 5.

$$\begin{aligned} P(A[i] \Rightarrow \hat{j}\,) &\sim \frac{P_{model}(\hat{A})}{P_{model}(A)} = \frac{P(i, \hat{j}, l)}{P(i, j, l)} \\ P(i, \hat{j}, l) &= \frac{\#(i, \hat{j}, l) + \alpha_2/D}{\#(i, *, l) + \alpha_2} \end{aligned} \tag{5}$$

Again, we assume a CRP when choosing a distortion value, where D is the expected number of distance values (set to 10 in our experiments), α_2 is the CRP hyperparameter, $\#(i, j, l)$ is the number of times a target word in position i was aligned to a source word in position j in sentences of length l, and $\#(i, *, l)$ is the number of times word in position i was aligned (to any source position) in sentences of length l.

Even without the need for handling different lengths for source and target sentences, this model is complex and requires estimating a separate probability for each triplet (i, j, l). In addition, the assumption that the distance distribution depends only on the sentence length and is similar for all tokens seems unreasonable, especially when dealing with part-of-speech tokens and dependency relations. Such concerns have been mentioned in the SMT literature and were shown to be justified in our experiments (see Sec. 4). For this reason, we

[3] Preliminary experiments using the EM approach via the GIZA++ toolkit (Och and Ney, 2003) resulted in similar performance to that of the sampling method for IBM Models 1 and 2. However, we were unable to explore the use of Model 3 under that framework, since the implementation of the model was strongly coupled to other, SMT-specific, optimizations and heuristics.

[4] Our implementation, as well as some core components in our framework, are based on code kindly provided by Chris Dyer.

also implemented an alternate distance model, based loosely on Liang et al. (2006). Under the alternate model, the probability of an alignment between target word i and source word $\hat{j}$ depends on the distance between them, their order, the sentence length, and the word type of the head, according to equation 6.

$$P(i,\hat{j},l) = \frac{\#[w_i,(i\text{-}\hat{j}),l] + \alpha_3/D}{\#(w_i,*,l)+\alpha_3} \tag{6}$$

IBM Model 3 This model handles the notion of fertility (or valence). Under this model, the probability of an alignment depends on how many target words are aligned to each of the source words. Each source word type $w_{\hat{j}}$, has a distribution specifying the probability of having n aligned target words. The probability of an alignment is proportional to the product of the probabilities of the fertilities in the alignment and takes into account the special status of the null word (represented by the index $j = 0$). This probability is given in Equation 7, which is based on Equation 32 in Brown et al. (1993)[5].

$$P(A) \sim \binom{l-\phi_0}{\phi_0} p_0^{l-2\phi_0} p_1^{\phi_0} \prod_{j=1}^{l} \phi_j! \frac{\#(w_j,\phi_j)+\alpha_4/F}{\#(w_j,*)+\alpha_4} \tag{7}$$

Here, ϕ_j denotes the number of target words aligned to the j-th source word in alignment A. p_1 and p_0 sum to 1 and are used to derive the probability that there will be ϕ_0 null-aligned words in a sentence containing l words[6]. $\#(w_j,\phi_j)$ represents the number of times source word w_j was observed to have ϕ_j dependent target words, $\#(w_j,*)$ is the number of times w_j appeared in the data, F is the expected number of fertility values (5 in our experiments), and α_4 is the CRP hyperparameter.

Combining the Models The original IBM models work in an incremental fashion, with each model using the output of the previous one as a starting point and adding a new component to the probability distribution. The dependency parsing framework employs a similar approach. It uses the alignments learned by the previous model as the starting point of the next and combines the probability distributions of each component via a product model. This allows for the easy introduction of new models which consider different aspects of the alignment and complement each other.

Preventing Self-Alignment When adapting the alignment approach to dependency parsing, we view the task as that of aligning a sentence to itself. One issue we must address is preventing the degenerate solution of aligning each word to itself. For this purpose we introduce a simple model into the product which gives zero probability to alignments which contain a word aligned to itself, as in equation 8.

$$P(A[i]=\hat{j}\,) = \begin{cases} 0 & \text{if } i = \hat{j} \\ \frac{1}{l-1} & \text{otherwise} \end{cases} \tag{8}$$

4 Experiments

4.1 Data

We evaluated our model on several corpora. The first of these was the Penn. Treebank portion of the Wall Street Journal (WSJ). We used the Constituent-to-Dependency Conversion Tool[7] to convert the treebank format into CoNLL format.

We also made use of the Danish and Dutch datasets from the CoNLL 2006 shared task[8]. Since we do not make use of annotation, we can induce a dependency structure on the entire dataset provided (disregarding the division into training and testing).

Following Klein and Manning (2004), we used the gold-standard part-of-speech sequences rather than the lexical forms and evaluated on sentences containing 10 or fewer tokens after removal of punctuation.

4.2 Results

Table 1 shows the results of the IBM Models on the task of directed (unlabeled) dependency parsing. We compare to the right-branching baseline used by Klein and Manning (2004). For the WSJ10 corpus, the authors reported 43.2% accuracy for DMV and 33.6% for the baseline. Although there are small

[5] The transitional version of this equation depends on whether either the old source word (j) or the new one ($\hat{j}$) are null, and is omitted for brevity. Further details can be found in Brown et al. (1993) Section 4.4 and Equation 43.

[6] For details, see Brown et al. (1993) Equation 31.

[7] `nlp.cs.lth.se/software/treebank_converter/`

[8] `http://nextens.uvt.nl/~conll/`

Corpus	M 1	M2	M3	R-br
WSJ10	25.42	35.73	39.32	32.85
Dutch10	25.17	32.46	35.28	28.42
Danish10	23.12	25.96	41.94	16.05 *

Table 1: Percent accuracy of IBM Models 1-3 (M1-3) and the right-branching baseline (R-br) on several corpora.

PoS	attachment	PoS	attachment
NN	DET	NNS	JJ
IN	NN	RB	VBZ
NNP	NNP	VBD	NN
DET	NN	VB	TO
JJ	NN	CC	NNS

Table 2: Most likely dependency attachment for the top ten most common parts-of-speech, according to Model 1.

differences in evaluation, as evidenced by the difference between our baseline scores, IBM Models 2 and 3 outperform the baseline by a large margin and Model 3 approaches the performance of DMV. On the Dutch and Danish datasets, the trends are similar. On the latter dataset, even Model 1 outperforms the right-branching baseline. However, the Danish dataset is unusual (see Buchholz and Marsi 2006) in that the alternate adjacency baseline of left-branching (also mentioned by Klein and Manning 2004) is extremely strong and achieves 48.8% directed accuracy.

4.3 Analysis

In order to better understand what our alignment model was learning, we looked at each component element individually.

Lexical Association To explore what Model 1 was learning, we analyzed the resulting probability tables for association between tokens. Table 2 shows the most likely dependency attachment for the top ten most common parts-of-speech. The model is clearly learning meaningful connections between parts of speech (determiners and adjectives to nouns, adverbs to verbs, etc.), but there is little notion of directionality, and cycles can exist. For instance, the model learns the connection between determiner and noun, but is unsure which is the head and which the dependent. A similar connection is learned between *to* and verbs in the base form (VB). This inconsistency is, to a large extent, the result of the deficiencies of the model, stemming from the fact that the IBM models were designed for a different task and are not trying to learn a well-formed tree. However, there is a strong linguistic basis to consider the directionality of these relations difficult. There is some debate among linguists as to whether the head of a noun phrase is the noun or the determiner[9] (see Abney 1987). Each can be seen as a different kind of head element, performing a different function, similarly to the multiple types of dependency relations identified in Hudson's (1990) Word Grammar. A similar case can be made regarding the head of an infinitive phrase. The infinitive form of the verb may be considered the lexical head, determining the predicate, while *to* can be seen as the functional head, encoding inflectional features, as in Chomsky's (1981) Government & Binding model[10].

Distance Models The original IBM distortion model (Model 2), which does not differentiate between words types and looks only at positions, has an accuracy of 33.43% on the WSJ10 corpus. In addition, it tends to strongly favor left-branching attachment (57.2% of target words were attached to the word immediately to their right, 22.6% to their left, as opposed to 31% and 25.8% in the gold standard). The alternative distance model we proposed, which takes into account the identity of the head word, achieves better accuracy and is closer to the gold standard balance (43.5% right and 35.3% left).

Figure 3 shows the distribution of the location of the dependent relative to the head word (at position 0) for several common parts-of-speech. It is interesting to see that singular and plural nouns (NN, NNS) behave similarly. They both have a strong preference for local attachment and a tendency towards a left-dependent (presumably the determiner, see above Table 2). Pronouns (NNP), on the other hand, are more likely to attach to the right since they are not modified by determiners. Verbs in past (VBZ) and present (VBD, VBP) forms have similar behavior, with a flatter distribution of dependent locations, whereas the base form (VB) attaches almost exclusively to the preceding token, presumably

[9] In fact, the original DMV chose the determiner as the head (see discussion in Klein and Manning 2004, Section 3).

[10] We thank an anonymous reviewer for elucidating this point.

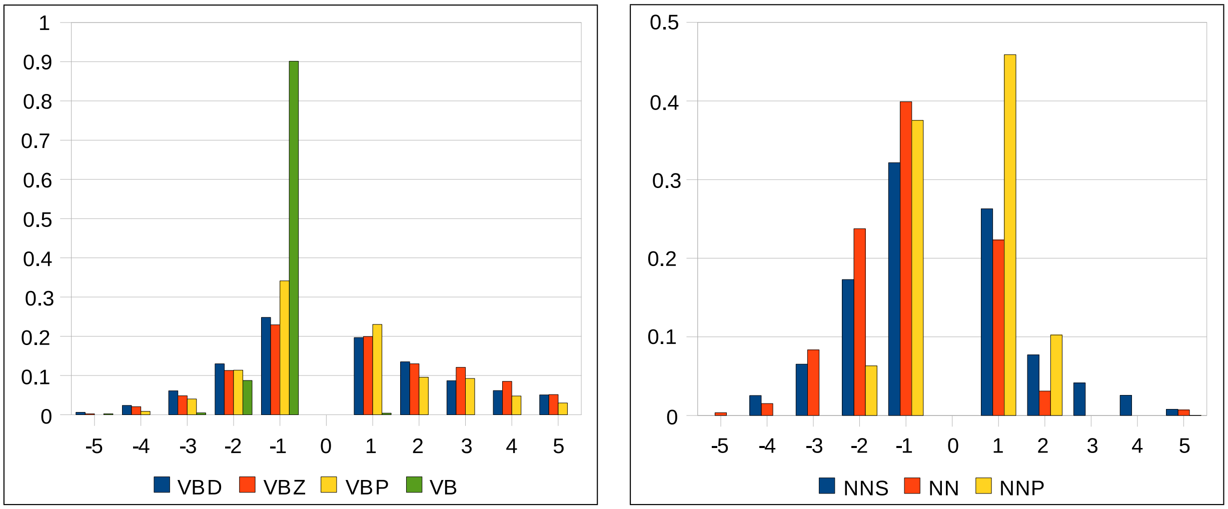


Figure 3: Distribution of head-to-dependent distance for several types of verbs (left) and nouns (right), as learned by our alternate distance model.

to (see Table 2).

Fertility Figure 4 shows the distribution of fertility values for several common parts of speech. Verbs have a relatively flat distribution with a longer tail as compared to nouns, which means they are likely to have a larger number of arguments. Once again, the base form (VB) exhibits different behavior from the other verbs forms, taking almost exclusively one argument. This is likely an effect of the strong connection between base form verbs and the preceding word *to*.

Hyper-Parameters Each of our models requires a value for its CRP hyperparameter (see Section 3.4). In this work, since parameter estimation was not our focus, we set the hyperparameters to be approximately $\frac{1}{K}$, where K is the number of possible values, according to the rule of thumb common in the literature. Specifically, we chose $\alpha_1 = 0.01, \alpha_3 = 0.05, \alpha_4 = 0.1$. We investigated the effect of these choices on performance in a separate set of experiments, which showed that small variations (up to an order of magnitude) in these parameters had little effect on the results.

In addition to the CRP parameters, Model 3 requires a value for p_1, the null fertility hyperparameter. In our experiments, we found that this hyperparameter had a very strong effect on results if it was above 0.1, creating many spurious null alignments. However, below that threshold, the effects were small. In the experiments reported here, we set $p_1 = 0.01$.

Initialization One issue with DMV, which is often mentioned, is its sensitivity to initialization. We tested our model with random initialization (uniform alignment probabilities) and with an approximation of the ad-hoc “harmonic” initialization described in Klein and Manning (2004) and found no noticeable difference in accuracy.

4.4 Discussion

The accuracy achieved by the IBM models (Table 1) is surprisingly high, given the fact that the IBM models were not designed with dependency parsing in mind. It is likely that customizing the models to the task will result in even better performance. Our findings in Section 4.3 support this hypothesis. The analysis showed that the lack of tree structure in the model impacted the learning, and therefore it is expected that a component which enforces tree structure (prevents cycles) will be beneficial.

Although it lacks an inherent notion of tree structure, the alignment-based approach has several advantages over the head-outward approach of DMV and related models. It can consider the alignment as a whole and take into account global sentence constraints, not just head-dependent relations. These may also include tree-structure constraints common to the head-outward approaches, but can be more flexible in how they are addressed. For instance,

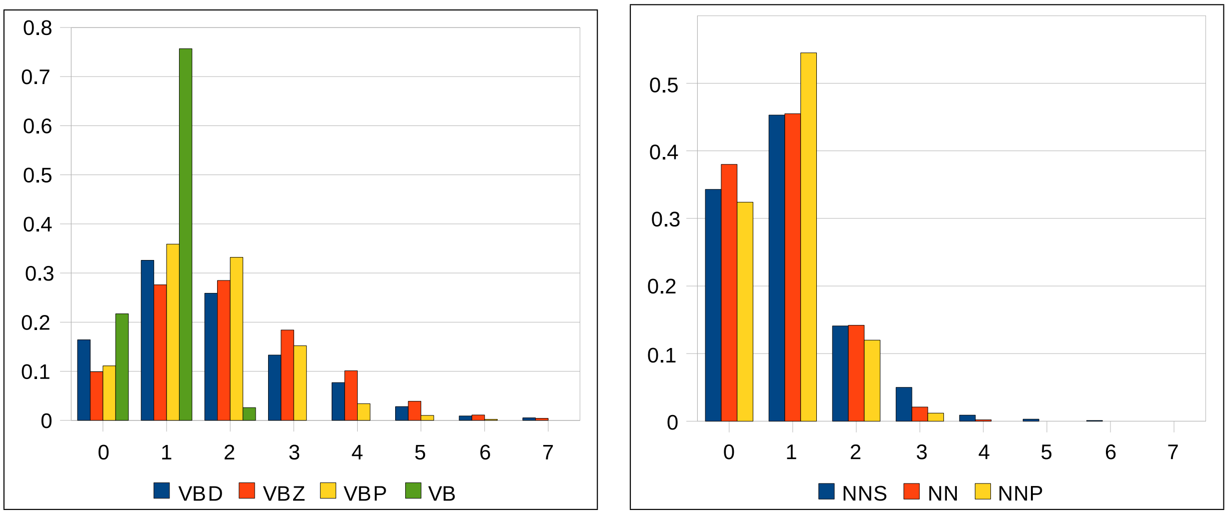


Figure 4: Distribution of fertility values for several types of verbs (left) and nouns (right), as learned by IBM Model 3.

DMV's method of modeling tree structure does not allow non-projective dependencies, whereas an alignment-based model may choose to allow or constrain non-projectivity, as learned from the data. Another advantage of our alignment-based models is the fact that they are not strongly sensitive to initialization and can be started from a set of random alignments.

5 Conclusions and Future Work

We have described an alternative formulation of dependency parsing as a problem of word alignment. This connection motivated us to explore the possibility of using alignment tools for the task of unsupervised dependency parsing. We chose to experiment with the well-known IBM alignment models which share a set of similar modeling assumptions with Klein and Manning's (2004) *Dependency Model with Valence*. Our experiments showed that the IBM models are surprisingly effective at the dependency parsing task, outperforming the right-branching baseline and approaching the accuracy of DMV. Our results demonstrate that the alignment approach can be used as a foundation for dependency parsing algorithms and motivates further research in this area.

There are many interesting avenues for further research. These include improving and extending the existing IBM models, as well as introducing new models that are specifically designed for the parsing task and represent relevant linguistic considerations (e.g., enforcing tree structure, handling crossing dependencies, learning left- or right-branching tendencies).

In Spitkovsky et al. (2010), the authors show that a gradual increase in the complexity of the data can aid the learning process. The IBM approach demonstrated the benefit of a gradual increase of *model* complexity. It would be interesting to see if the two approaches could be successfully combined.

Finally, although we use our framework for dependency parsing, the sampling approach and the framework we developed can be used to explore new models for bilingual word alignment. Furthermore, an alignment-based parsing method is expected to integrate well with SMT bi-lingual alignment models and may, therefore, be suitable for combined models which use parse trees to improve word alignment (e.g., Burkett et al. 2010).

Acknowledgments

I would like to thank Chris Dyer for providing the basis for the sampling implementation. I would also like to thank Chris, Adam Lopez, Trevor Cohn, Adam Faulkner and the anonymous reviewers for their time and effort and their helpful comments and suggestions.

References

Abney, Steven. 1987. *The English Noun Phrase in its Sentential Aspect*. Ph.D. thesis, Massachusetts Institute of Technology.

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

Buchholz, Sabine and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *In Proc. of CoNLL*. pages 149–164.

Burkett, David, John Blitzer, and Dan Klein. 2010. Joint parsing and alignment with weakly synchronized grammars. In *North American Association for Computational Linguistics*. Los Angeles.

Chiang, David. 2005. A hierarchical phrase-based model for statistical machine translation. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Morristown, NJ, USA, pages 263–270.

Chomsky, Noam. 1981. *Lectures on government and binding : the Pisa lectures / Noam Chomsky*.

Cohen, Shay B. and Noah A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *NAACL '09: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Morristown, NJ, USA, pages 74–82.

Haghighi, Aria and Dan Klein. 2010. Coreference resolution in a modular, entity-centered model. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Los Angeles, California, pages 385–393.

Headden III, William P., Mark Johnson, and David McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Boulder, Colorado, pages 101–109.

Hudson, R. 1990. *English Word Grammar*. Basil Blackwell, Oxford.

Klein, Dan and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: models of dependency and constituency. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Morristown, NJ, USA, page 478.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *ACL '07: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. Association for Computational Linguistics, Morristown, NJ, USA, pages 177–180.

Liang, Percy, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*. Association for Computational Linguistics, New York City, USA, pages 104–111.

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

Paskin, Mark A. 2001. Grammatical bigrams. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *NIPS*. MIT Press, pages 91–97.

Spitkovsky, Valentin I., Hiyan Alshawi, and Daniel Jurafsky. 2010. From Baby Steps to Leapfrog: How "Less is More" in unsupervised dependency parsing. In *Proc. of NAACL-HLT*.

Yuret, D. 1998. *Discovery of linguistic relations using lexical attraction*. Ph.D. thesis, Department of Computer Science and Electrical Engineering, MIT.

Inducing Probabilistic CCG Grammars from Logical Form with Higher-Order Unification

Tom Kwiatkowski[*] t.m.kwiatkowksi@sms.ed.ac.uk
Luke Zettlemoyer[†] lsz@cs.washington.edu
Sharon Goldwater[*] sgwater@inf.ed.ac.uk
Mark Steedman[*] steedman@inf.ed.ac.uk

[*]School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK

[†]Computer Science & Engineering
University of Washington
Seattle, WA 98195

Abstract

This paper addresses the problem of learning to map sentences to logical form, given training data consisting of natural language sentences paired with logical representations of their meaning. Previous approaches have been designed for particular natural languages or specific meaning representations; here we present a more general method. The approach induces a probabilistic CCG grammar that represents the meaning of individual words and defines how these meanings can be combined to analyze complete sentences. We use higher-order unification to define a hypothesis space containing all grammars consistent with the training data, and develop an online learning algorithm that efficiently searches this space while simultaneously estimating the parameters of a log-linear parsing model. Experiments demonstrate high accuracy on benchmark data sets in four languages with two different meaning representations.

1 Introduction

A key aim in natural language processing is to learn a mapping from natural language sentences to formal representations of their meaning. Recent work has addressed this problem by learning semantic parsers given sentences paired with logical meaning representations (Thompson & Mooney, 2002; Kate et al., 2005; Kate & Mooney, 2006; Wong & Mooney, 2006, 2007; Zettlemoyer & Collins, 2005, 2007; Lu et al., 2008). For example, the training data might consist of English sentences paired with lambda-calculus meaning representations:

Sentence: which states border texas
Meaning: $\lambda x.state(x) \wedge next\_to(x, tex)$

Given pairs like this, the goal is to learn to map new, unseen, sentences to their corresponding meaning.

Previous approaches to this problem have been tailored to specific natural languages, specific meaning representations, or both. Here, we develop an approach that can learn to map any natural language to a wide variety of logical representations of linguistic meaning. In addition to data like the above, this approach can also learn from examples such as:

Sentence: hangi eyaletin texas ye siniri vardir
Meaning: $answer(state(borders(tex)))$

where the sentence is in Turkish and the meaning representation is a variable-free logical expression of the type that has been used in recent work (Kate et al., 2005; Kate & Mooney, 2006; Wong & Mooney, 2006; Lu et al., 2008).

The reason for generalizing to multiple languages is obvious. The need to learn over multiple representations arises from the fact that there is no standard representation for logical form for natural language. Instead, existing representations are ad hoc, tailored to the application of interest. For example, the variable-free representation above was designed for building natural language interfaces to databases.

Our approach works by inducing a combinatory categorial grammar (CCG) (Steedman, 1996, 2000). A CCG grammar consists of a language-specific lexicon, whose entries pair individual words and phrases with both syntactic and semantic information, and a universal set of combinatory rules that

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1223–1233,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

project that lexicon onto the sentences and meanings of the language via syntactic derivations. The learning process starts by postulating, for each sentence in the training data, a single multi-word lexical item pairing that sentence with its complete logical form. These entries are iteratively refined with a restricted higher-order unification procedure (Huet, 1975) that defines all possible ways to subdivide them, consistent with the requirement that each training sentence can still be parsed to yield its labeled meaning.

For the data sets we consider, the space of possible grammars is too large to explicitly enumerate. The induced grammar is also typically highly ambiguous, producing a large number of possible analyses for each sentence. Our approach discriminates between analyses using a log-linear CCG parsing model, similar to those used in previous work (Clark & Curran, 2003, 2007), but differing in that the syntactic parses are treated as a hidden variable during training, following the approach of Zettlemoyer & Collins (2005, 2007). We present an algorithm that incrementally learns the parameters of this model while simultaneously exploring the space of possible grammars. The model is used to guide the process of grammar refinement during training as well as providing a metric for selecting the best analysis for each new sentence.

We evaluate the approach on benchmark datasets from a natural language interface to a database of US Geography (Zelle & Mooney, 1996). We show that accurate models can be learned for multiple languages with both the variable-free and lambda-calculus meaning representations introduced above. We also compare performance to previous methods (Kate & Mooney, 2006; Wong & Mooney, 2006, 2007; Zettlemoyer & Collins, 2005, 2007; Lu et al., 2008), which are designed with either language- or representation- specific constraints that limit generalization, as discussed in more detail in Section 6. Despite being the only approach that is general enough to run on all of the data sets, our algorithm achieves similar performance to the others, even outperforming them in several cases.

2 Overview of the Approach

The goal of our algorithm is to find a function $f : x \rightarrow z$ that maps sentences x to logical expressions z. We learn this function by inducing a probabilistic CCG (PCCG) grammar from a training set $\{(x_i, z_i)|i = 1 \ldots n\}$ containing example (sentence, logical-form) pairs such as ("New York borders Vermont", $next\_to(ny, vt)$). The induced grammar consists of two components which the algorithm must learn:

- A CCG lexicon, Λ, containing lexical items that define the space of possible parses y for an input sentence x. Each parse contains both syntactic and semantic information, and defines the output logical form z.
- A parameter vector, θ, that defines a distribution over the possible parses y, conditioned on the sentence x.

We will present the approach in two parts. The lexical induction process (Section 4) uses a restricted form of higher order unification along with the CCG combinatory rules to propose new entries for Λ. The complete learning algorithm (Section 5) integrates this lexical induction with a parameter estimation scheme that learns θ. Before presenting the details, we first review necessary background.

3 Background

This section provides an introduction to the ways in which we will use lambda calculus and higher-order unification to construct meaning representations. It also reviews the CCG grammar formalism and probabilistic extensions to it, including existing parsing and parameter estimation techniques.

3.1 Lambda Calculus and Higher-Order Unification

We assume that sentence meanings are represented as logical expressions, which we will construct from the meaning of individual words by using the operations defined in the lambda calculus. We use a version of the typed lambda calculus (cf. Carpenter (1997)), in which the basic types include e, for entities; t, for truth values; and i for numbers. There are also function types of the form $\langle e, t\rangle$ that are assigned to lambda expressions, such as $\lambda x.state(x)$, which take entities and return truth values. We represent the meaning of words and phrases using lambda-calculus expressions that can contain constants, quantifiers, logical connectors, and lambda abstractions.

The advantage of using the lambda calculus lies in its generality. The meanings of individual words and phrases can be arbitrary lambda expressions, while the final meaning for a sentence can take different forms. It can be a full lambda-calculus expression, a variable-free expression such as $answer(state(borders(tex)))$, or any other logical expression that can be built from the primitive meanings via function application and composition.

The higher-order unification problem (Huet, 1975) involves finding a substitution for the free variables in a pair of lambda-calculus expressions that, when applied, makes the expressions equal each other. This problem is notoriously complex; in the unrestricted form (Huet, 1973), it is undecidable. In this paper, we will guide the grammar induction process using a restricted version of higher-order unification that is tractable. For a given expression h, we will need to find expressions for f and g such that either $h = f(g)$ or $h = \lambda x.f(g(x))$. This limited form of the unification problem will allow us to define the ways to split h into subparts that can be recombined with CCG parsing operations, which we will define in the next section, to reconstruct h.

3.2 Combinatory Categorial Grammar

CCG (Steedman, 2000) is a linguistic formalism that tightly couples syntax and semantics, and can be used to model a wide range of language phenomena. For present purposes a CCG grammar includes a lexicon Λ with entries like the following:

$$\begin{array}{l}
\text{New York} \vdash NP : ny \\
\text{borders} \vdash S\backslash NP/NP : \lambda x\lambda y.next\_to(y,x) \\
\text{Vermont} \vdash NP : vt
\end{array}$$

where each lexical item $w \vdash X : h$ has words w, a syntactic category X, and a logical form h expressed as a lambda-calculus expression. For the first example, these are "New York," NP, and ny. CCG syntactic categories may be atomic (such as S, NP) or complex (such as $S\backslash NP/NP$).

CCG combines categories using a set of *combinatory rules*. For example, the forward ($>$) and backward ($<$) *application* rules are:

$$\begin{array}{lclr}
X/Y : f \quad Y : g & \Rightarrow & X : f(g) & (>) \\
Y : g \quad X\backslash Y : f & \Rightarrow & X : f(g) & (<)
\end{array}$$

These rules apply to build syntactic and semantic derivations under the control of the word order information encoded in the slash directions of the lexical entries. For example, given the lexicon above, the sentence *New York borders Vermont* can be parsed to produce:

$$\begin{array}{cccl}
\text{New York} & \text{borders} & \text{Vermont} & \\
\cline{1-1}\cline{2-2}\cline{3-3}
NP & (S\backslash NP)/NP & NP & \\
ny & \lambda x\lambda y.next\_to(y,x) & vt & \\
\cline{2-3}
 & \multicolumn{2}{c}{(S\backslash NP)} & > \\
 & \multicolumn{2}{c}{\lambda y.next\_to(y,vt)} & \\
\cline{1-3}
\multicolumn{3}{c}{S} & < \\
\multicolumn{3}{c}{next\_to(ny,vt)} &
\end{array}$$

where each step in the parse is labeled with the combinatory rule ($- >$ or $- <$) that was used.

CCG also includes combinatory rules of forward ($>$ **B**) and backward ($<$ **B**) *composition*:

$$\begin{array}{llr}
X/Y : f \quad Y/Z : g & \Rightarrow X/Z : \lambda x.f(g(x)) & (> \mathbf{B}) \\
Y\backslash Z : g \quad X\backslash Y : f & \Rightarrow X\backslash Z : \lambda x.f(g(x)) & (< \mathbf{B})
\end{array}$$

These rules provide for a relaxed notion of constituency which will be useful during learning as we reason about possible refinements of the grammar.

We also allow vertical slashes in CCG categories, which act as wild cards. For example, with this extension the forward application combinator ($>$) could be used to combine the category $S/(S|NP)$ with any of $S\backslash NP$, S/NP, or $S|NP$. Figure 1 shows two parses where the composition combinators and vertical slashes are used. These parses closely resemble the types of analyses that will be possible under the grammars we learn in the experiments described in Section 8.

3.3 Probabilistic CCGs

Given a CCG lexicon Λ, there will, in general, be many possible parses for each sentence. We select the most likely alternative using a log-linear model, which consists of a feature vector ϕ and a parameter vector θ. The joint probability of a logical form z constructed with a parse y, given a sentence x is

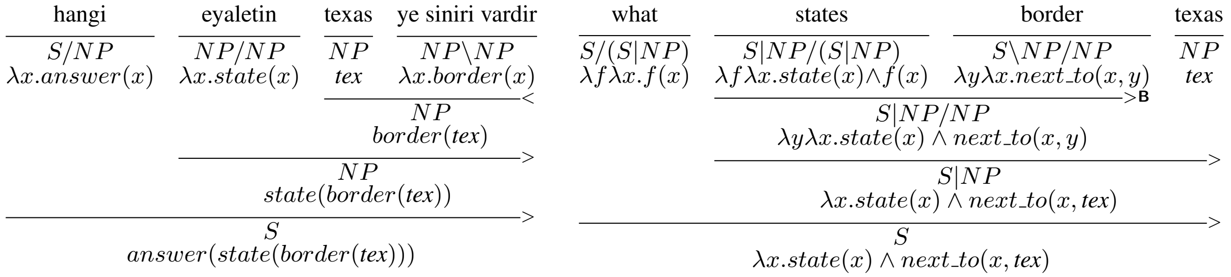


Figure 1: Two examples of CCG parses with different logical form representations.

defined as:

$$P(y,z|x;\theta,\Lambda) = \frac{e^{\theta\cdot\phi(x,y,z)}}{\sum_{(y',z')} e^{\theta\cdot\phi(x,y',z')}} \qquad (1)$$

Section 7 defines the features used in the experiments, which include, for example, lexical features that indicate when specific lexical items in Λ are used in the parse y. For parsing and parameter estimation, we use standard algorithms (Clark & Curran, 2007), as described below.

The parsing, or inference, problem is to find the most likely logical form z given a sentence x, assuming the parameters θ and lexicon Λ are known:

$$f(x) = \arg\max_z p(z|x;\theta,\Lambda) \qquad (2)$$

where the probability of the logical form is found by summing over all parses that produce it:

$$p(z|x;\theta,\Lambda) = \sum_y p(y,z|x;\theta,\Lambda) \qquad (3)$$

In this approach the distribution over parse trees y is modeled as a hidden variable. The sum over parses in Eq. 3 can be calculated efficiently using the inside-outside algorithm with a CKY-style parsing algorithm.

To estimate the parameters themselves, we use stochastic gradient updates (LeCun et al., 1998). Given a set of n sentence-meaning pairs $\{(x_i,z_i) : i = 1...n\}$, we update the parameters θ iteratively, for each example i, by following the local gradient of the conditional log-likelihood objective $O_i = \log P(z_i|x_i;\theta,\Lambda)$. The local gradient of the individual parameter θ_j associated with feature ϕ_j and training instance (x_i,z_i) is given by:

$$\begin{aligned}\frac{\partial O_i}{\partial \theta_j} &= E_{p(y|x_i,z_i;\theta,\Lambda)}[\phi_j(x_i,y,z_i)] \\ &\quad -E_{p(y,z|x_i;\theta,\Lambda)}[\phi_j(x_i,y,z)]\end{aligned} \qquad (4)$$

As with Eq. 3, all of the expectations in Eq. 4 are calculated through the use of the inside-outside algorithm on a pruned parse chart. In the experiments, each chart cell was pruned to the top 200 entries.

4 Splitting Lexical Items

Before presenting a complete learning algorithm, we first describe how to use higher-order unification to define a procedure for splitting CCG lexical entries. This splitting process is used to expand the lexicon during learning. We seed the lexical induction with a multi-word lexical item $x_i \vdash S{:}z_i$ for each training example (x_i,z_i), consisting of the entire sentence x_i and its associated meaning representation z_i. For example, one initial lexical item might be:

$$\text{New York borders Vermont} \vdash S{:}next\_to(ny,vt) \qquad (5)$$

Although these initial, sentential lexical items can parse the training data, they will not generalize well to unseen data. To learn effectively, we will need to split overly specific entries of this type into pairs of new, smaller, entries that generalize better. For example, one possible split of the lexical entry given in (5) would be the pair:

$$\text{New York borders} \vdash S/NP : \lambda x.next\_to(ny,x),$$
$$\text{Vermont} \vdash NP : vt$$

where we broke the original logical expression into two new ones $\lambda x.next\_to(ny,x)$ and vt, and paired them with syntactic categories that allow the new lexical entries to be recombined to produce the original analysis. The next three subsections define the set of possible splits for any given lexical item. The process is driven by solving a higher-order unification problem that defines all of the ways of splitting the logical expression into two parts, as described in Section 4.1. Section 4.2 describes how to construct syntactic categories that are consistent with the two new fragments of logical form and which will allow the new lexical items to recombine. Finally, Section 4.3 defines the full set of lexical entry pairs that can be created by splitting a lexical entry.

As we will see, this splitting process is overly prolific for any single language and will yield many lexical items that do not generalize well. For example, there is nothing in our original lexical entry above that provides evidence that the split should pair "Vermont" with the constant vt and not $\lambda x.next\_to(ny, x)$. Section 5 describes how we estimate the parameters of a probabilistic parsing model and how this parsing model can be used to guide the selection of items to add to the lexicon.

4.1 Restricted Higher-Order Unification

The set of possible splits for a logical expression h is defined as the solution to a pair of higher-order unification problems. We find pairs of logical expressions (f, g) such that either $f(g) = h$ or $\lambda x.f(g(x)) = h$. Solving these problems creates new expressions f and g that can be recombined according to the CCG combinators, as defined in Section 3.2, to produce h.

In the unrestricted case, there can be infinitely many solution pairs (f, g) for a given expression h. For example, when $h = tex$ and $f = \lambda x.tex$, the expression g can be anything. Although it would be simple enough to forbid vacuous variables in f and g, the number of solutions would still be exponential in the size of h. For example, when h contains a conjunction, such as $h = \lambda x.city(x) \wedge major(x) \wedge in(x, tex)$, any subset of the expressions in the conjunction can be assigned to f (or g).

To limit the number of possible splits, we enforce the following restrictions on the possible higher-order solutions that will be used during learning:

- **No Vacuous Variables:** Neither g or f can be a function of the form $\lambda x.e$ where the expression e does not contain the variable x. This rules out functions such as $\lambda x.tex$.
- **Limited Coordination Extraction:** The expression g cannot contain more than N of the conjuncts that appear in any coordination in h. For example, with $N = 1$ the expression $g = \lambda x.city(x) \wedge major(x)$ could not be used as a solution given the h conjuction above. We use $N = 4$ in our experimental evaluation.
- **Limited Application:** The function f cannot contain new variables applied to any non-variable subexpressions from h. For example, if $h = \lambda x.in(x, tex)$, the pair $f = \lambda q.q(tex)$ and $g = \lambda y\lambda x.in(x, y)$ is forbidden.

Together, these three restrictions guarantee that the number of splits is, in the worst case, an N-degree polynomial of the number of constants in h. The constraints were designed to increase the efficiency of the splitting algorithm without impacting performance on the development data.

4.2 Splitting Categories

We define the set of possible splits for a category $X : h$ with syntax X and logical form h by enumerating the solution pairs (f, g) to the higher-order unification problems defined above and creating syntactic categories for the resulting expressions. For example, given $X : h = S\backslash NP : \lambda x.in(x, tex)$, $f = \lambda y\lambda x.in(x, y)$, and $g = tex$, we would produce the following two pairs of new categories:

$$(S\backslash NP/NP{:}\lambda y\lambda x.in(x, y) \ , \ NP{:}tex)$$
$$(NP{:}tex \ , \ S\backslash NP\backslash NP{:}\lambda y\lambda x.in(x, y))$$

which were constructed by first choosing the syntactic category for g, in this case NP, and then enumerating the possible directions for the new slash in the category containing f. We consider each of these two steps in more detail below.

The new syntactic category for g is determined based on its type, $T(g)$. For example, $T(tex) = e$ and $T(\lambda x.state(x)) = \langle e, t\rangle$. Then, the function $C(T)$ takes an input type T and returns the syntactic category of T as follows:

$$C(T) = \begin{cases} NP & \text{if } T = e \\ S & \text{if } T = t \\ C(T_2)|C(T_1) & \text{when } T = \langle T_1, T_2\rangle \end{cases}$$

The basic types e and t are assigned syntactic categories NP and S, and all functional types are assigned categories recursively. For example $C(\langle e, t\rangle) = S|NP$ and $C(\langle e, \langle e, t\rangle\rangle) = S|NP|NP$. This definition of CCG categories is unconventional in that it never assigns atomic categories to functional types. For example, there is no distinct syntactic category N for nouns (which have semantic type $\langle e, t\rangle$). Instead, the more complex category $S|NP$ is used.

Now, we are ready to define the set of all category splits. For a category $A = X{:}h$ we can define

$$\mathrm{S}_C(A) = \{\mathrm{FA}(A) \cup \mathrm{BA}(A) \cup \mathrm{FC}(A) \cup \mathrm{BC}(A)\}$$

which is a union of sets, each of which includes splits for a single CCG operator. For example, FA$(X{:}h)$ is the set of category pairs

$$\mathrm{FA}(X{:}h) = \{(X/Y{:}f, Y{:}g) \mid h{=}f(g) \wedge Y{=}C(T(g))\}$$

where each pair can be combined with the forward application combinator, described in Section 3.2, to reconstruct $X{:}h$.

The remaining three sets are defined similarly, and are associated with the backward application and forward and backward composition operators, respectively:

$$\mathrm{BA}(X{:}h) = \{(Y{:}g, X\backslash Y{:}f) \mid h{=}f(g) \wedge Y{=}C(T(g))\}$$

$$\mathrm{FC}(X/Y{:}h) = \{(X/W{:}f, W/Y{:}g) \mid h{=}\lambda x.f(g(x)) \wedge W{=}C(T(g(x)))\}$$

$$\mathrm{BC}(X\backslash Y{:}h) = \{(W\backslash Y{:}g, X\backslash W{:}f) \mid h{=}\lambda x.f(g(x)) \wedge W{=}C(T(g(x)))\}$$

where the composition sets FC and BC only accept input categories with the appropriate outermost slash direction, for example FC$(X/Y{:}h)$.

4.3 Splitting Lexical Items

We can now define the lexical splits that will be used during learning. For lexical entry $w_{0:n} \vdash A$, with word sequence $w_{0:n} = \langle w_0, \ldots, w_n\rangle$ and CCG category A, define the set S_L of splits to be:

$$\mathrm{S}_L(w_{0:n}{\vdash}A) = \{(w_{0:i}{\vdash}B, w_{i+1:n}{\vdash}C) \mid 0 \leq i < n \wedge (B, C) \in \mathrm{S}_C(A)\}$$

where we enumerate all ways of splitting the words sequence $w_{0:n}$ and aligning the subsequences with categories in $\mathrm{S}_C(A)$, as defined in the last section.

5 Learning Algorithm

The previous section described how a splitting procedure can be used to break apart overly specific lexical items into smaller ones that may generalize better to unseen data. The space of possible lexical items supported by this splitting procedure is too large to explicitly enumerate. Instead, we learn the parameters of a PCCG, which is used both to guide the splitting process, and also to select the best parse, given a learned lexicon.

Figure 2 presents the unification-based learning algorithm, UBL. This algorithm steps through the data incrementally and performs two steps for each training example. First, new lexical items are induced for the training instance by splitting and merging nodes in the best correct parse, given the current parameters. Next, the parameters of the PCCG are updated by making a stochastic gradient update on the marginal likelihood, given the updated lexicon.

Inputs and Initialization The algorithm takes as input the training set of n (sentence, logical form) pairs $\{(x_i, z_i) : i = 1...n\}$ along with an NP list, Λ_{NP}, of proper noun lexical items such as $\mathrm{Texas} \vdash NP{:}tex$. The lexicon, Λ, is initialized with a single lexical item $x_i \vdash S : z_i$ for each of the training pairs along with the contents of the NP list. It is possible to run the algorithm without the initial NP list; we include it to allow direct comparisons with previous approaches, which also included NP lists. Features and initial feature weights are described in Section 7.

Step 1: Updating the Lexicon In the lexical update step the algorithm first computes the best correct parse tree y^* for the current training example and then uses y^* as input to the procedure NEW-LEX, which determines which (if any) new lexical items to add to Λ. NEW-LEX begins by enumerating all pairs $(C, w_{i:j})$, for $i < j$, where C is a category occurring at a node in y^* and $w_{i:j}$ are the (two or more) words it spans. For example, in the left parse in Figure 1, there would be four pairs: one with the category $C = NP\backslash NP{:}\lambda x.border(x)$ and the phrase $w_{i:j}$ ="ye siniri vardir", and one for each non-leaf node in the tree.

For each pair $(C, w_{i:j})$, NEW-LEX considers introducing a new lexical item $w_{i:j} {\vdash} C$, which allows for the possibility of a parse where the subtree rooted at C is replaced with this new entry. (If C is a leaf node, this item will already exist.) NEW-LEX also considers adding each pair of new lexical items that is obtained by splitting $w_{i:j}{\vdash}C$ as described in Section 4, thereby considering many different ways of reanalyzing the node. This process creates a set of possible new lexicons, where each lexicon expands

Λ in a different way by adding the items from either a single split or a single merge of a node in y^*.

For each potential new lexicon Λ', NEW-LEX computes the probability $p(y^*|x_i, z_i; \theta', \Lambda')$ of the original parse y^* under Λ' and parameters θ' that are the same as θ but have weights for the new lexical items, as described in Section 7. It also finds the best new parse $y' = \arg\max_y p(y|x_i, z_i; \theta', \Lambda')$.[1] Finally, NEW-LEX selects the Λ' with the largest difference in log probability between y' and y^*, and returns the new entries in Λ'. If y^* is the best parse for every Λ', NEW-LEX returns the empty set; the lexicon will not change.

Step 2: Parameter Updates For each training example we update the parameters θ using the stochastic gradient updates given by Eq. 4.

Discussion The alternation between refining the lexicon and updating the parameters drives the learning process. The initial model assigns a conditional likelihood of one to each training example (there is a single lexical item for each sentence x_i, and it contains the labeled logical form z_i). Although the splitting step often decreases the probability of the data, the new entries it produces are less specific and should generalize better. Since we initially assign positive weights to the parameters for new lexical items, the overall approach prefers splitting; trees with many lexical items will initially be much more likely. However, if the learned lexical items are used in too many incorrect parses, the stochastic gradient updates will down weight them to the point where the lexical induction step can merge or re-split nodes in the trees that contain them. This allows the approach to correct the lexicon and, hopefully, improve future performance.

```
Inputs: Training set $\{(x_i, z_i) : i = 1 \ldots n\}$ where each
    example is a sentence $x_i$ paired with a logical form
    $z_i$. Set of NP lexical items $\Lambda_{NP}$. Number of iterations $T$. Learning rate parameter $\alpha_0$ and cooling
    rate parameter $c$.
Definitions: The function NEW-LEX($y$) takes a parse
    $y$ and returns a set of new lexical items found by
    splitting and merging categories in $y$, as described
    in Section 5. The distributions $p(y|x, z; \theta, \Lambda)$ and
    $p(y, z|x; \theta, \Lambda)$ are defined by the log-linear model,
    as described in Section 3.3.
Initialization:
  • Set $\Lambda = \{x_i \vdash S : z_i\}$ for all $i = 1 \ldots n$.
  • Set $\Lambda = \Lambda \cup \Lambda_{NP}$
  • Initialize $\theta$ using coocurrence statistics, as described in Section 7.
Algorithm:
    For $t = 1 \ldots T, i = 1 \ldots n$ :
    Step 1: (Update Lexicon)
        • Let $y^* = \arg\max_y p(y|x_i, z_i; \theta, \Lambda)$
        • Set $\Lambda = \Lambda \cup$ NEW-LEX($y^*$) and expand the
          parameter vector $\theta$ to contain entries for the
          new lexical items, as described in Section 7.
    Step 2: (Update Parameters)
        • Let $\gamma = \frac{\alpha_0}{1+c\times k}$ where $k = i + t \times n$.
        • Let $\Delta = E_{p(y|x_i,z_i;\theta,\Lambda)}[\phi(x_i, y, z_i)]$
                  $-E_{p(y,z|x_i;\theta,\Lambda)}[\phi(x_i, y, z)]$
        • Set $\theta = \theta + \gamma\Delta$
Output: Lexicon $\Lambda$ and parameters $\theta$.
```

Figure 2: The UBL learning algorithm.

6 Related Work

Previous work has focused on a variety of different meaning representations. Several approaches have been designed for the variable-free logical representations shown in examples throughout this paper. For example, Kate & Mooney (2006) present a method (KRISP) that extends an existing SVM learning algorithm to recover logical representations. The WASP system (Wong & Mooney, 2006) uses statistical machine translation techniques to learn synchronous context free grammars containing both words and logic. Lu et al. (2008) (Lu08) developed a generative model that builds a single hybrid tree of words, syntax and meaning representation. These algorithms are all language independent but representation specific.

Other algorithms have been designed to recover lambda-calculus representations. For example, Wong & Mooney (2007) developed a variant of WASP (λ-WASP) specifically designed for this alternate representation. Zettlemoyer & Collins (2005, 2007) developed CCG grammar induction techniques where lexical items are proposed according to a set of hand-engineered lexical templates.

[1] This computation can be performed efficiently by incrementally updating the parse chart used to find y^*.

Our approach eliminates this need for manual effort.

Another line of work has focused on recovering meaning representations that are not based on logic. Examples include an early statistical method for learning to fill slot-value representations (Miller et al., 1996) and a more recent approach for recovering semantic parse trees (Ge & Mooney, 2006). Exploring the extent to which these representations are compatible with the logic-based learning approach we developed is an important area for future work.

Finally, there is work on using categorial grammars to solve other, related learning problems. For example, Buszkowski & Penn (1990) describe a unification-based approach for grammar discovery from bracketed natural language sentences and Villavicencio (2002) developed an approach for modeling child language acquisition. Additionally, Bos et al. (2004) consider the challenging problem of constructing broad-coverage semantic representations with CCG, but do not learn the lexicon.

7 Experimental Setup

Features We use two types of features in our model. First, we include a set of *lexical features*: For each lexical item $L \in \Lambda$, we include a feature ϕ_L that fires when L is used. Second, we include *semantic features* that are functions of the output logical expression z. Each time a predicate p in z takes an argument a with type $T(a)$ in position i it triggers two binary indicator features: $\phi_{(p,a,i)}$ for the predicate-argument relation; and $\phi_{(p,T(a),i)}$ for the predicate argument-type relation.

Initialization The weights for the semantic features are initialized to zero. The weights for the lexical features are initialized according to coocurrance statistics estimated with the Giza++ (Och & Ney, 2003) implementation of IBM Model 1. We compute translation scores for (word, constant) pairs that cooccur in examples in the training data. The initial weight for each ϕ_L is set to ten times the average score over the (word, constant) pairs in L, except for the weights of seed lexical entries in Λ_{NP} which are set to 10 (equivalent to the highest possible coocurrence score). We used the learning rate $\alpha_0 = 1.0$ and cooling rate $c = 10^{-5}$ in all training scenarios, and ran the algorithm for $T = 20$ iterations. These values were selected with cross validation on the Geo880 development set, described below.

Data and Evaluation We evaluate our system on the GeoQuery datasets, which contain natural-language queries of a geographical database paired with logical representations of each query's meaning. The full Geo880 dataset contains 880 (English-sentence, logical-form) pairs, which we split into a development set of 600 pairs and a test set of 280 pairs, following Zettlemoyer & Collins (2005). The Geo250 dataset is a subset of Geo880 containing 250 sentences that have been translated into Turkish, Spanish and Japanese as well as the original English. Due to the small size of this dataset we use 10-fold cross validation for evaluation. We use the same folds as Wong & Mooney (2006, 2007) and Lu et al. (2008), allowing a direct comparison.

The GeoQuery data is annotated with both lambda-calculus and variable-free meaning representations, which we have seen examples of throughout the paper. We report results for both representations, using the standard measures of *Recall* (percentage of test sentences assigned correct logical forms), *Precision* (percentage of logical forms returned that are correct) and *F1* (the harmonic mean of *Precision* and *Recall*).

Two-Pass Parsing To investigate the trade-off between precision and recall, we report results with a two-pass parsing strategy. When the parser fails to return an analysis for a test sentence due to novel words or usage, we reparse the sentence and allow the parser to skip words, with a fixed cost. Skipping words can potentially increase recall, if the ignored word is an unknown function word that does not contribute semantic content.

8 Results and Discussion

Tables 1, 2, and 3 present the results for all of the experiments. In aggregate, they demonstrate that our algorithm, UBL, learns accurate models across languages and for both meaning representations. This is a new result; no previous system is as general.

We also see the expected tradeoff between precision and recall that comes from the two-pass parsing approach, which is labeled UBL-s. With the ability to skip words, UBL-s achieves the highest recall of all reported systems for all evaluation conditions.

System	English			Spanish		
	Rec.	Pre.	F1	Rec.	Pre.	F1
WASP	70.0	**95.4**	80.8	72.4	91.2	81.0
Lu08	72.8	91.5	81.1	79.2	**95.2**	**86.5**
UBL	78.1	88.2	**82.7**	76.8	86.8	81.4
UBL-s	**80.4**	80.8	80.6	**79.7**	80.6	80.1
System	Japanese			Turkish		
	Rec.	Pre.	F1	Rec.	Pre.	F1
WASP	74.4	**92.0**	**82.9**	62.4	**97.0**	75.9
Lu08	76.0	87.6	81.4	66.8	93.8	78.0
UBL	78.5	85.5	81.8	70.4	89.4	**78.6**
UBL-s	**80.5**	80.6	80.6	**74.2**	75.6	74.9

Table 1: Performance across languages on Geo250 with variable-free meaning representations.

System	English			Spanish		
	Rec.	Pre.	F1	Rec.	Pre.	F1
λ-WASP	75.6	91.8	82.9	80.0	92.5	**85.8**
UBL	78.0	**93.2**	**84.7**	75.9	**93.4**	83.6
UBL-s	**81.8**	83.5	82.6	**81.4**	83.4	82.4
System	Japanese			Turkish		
	Rec.	Pre.	F1	Rec.	Pre.	F1
λ-WASP	81.2	90.1	**85.8**	68.8	90.4	**78.1**
UBL	78.9	**90.9**	84.4	67.4	**93.4**	**78.1**
UBL-s	**83.0**	83.2	83.1	**71.8**	77.8	74.6

Table 2: Performance across languages on Geo250 with lambda-calculus meaning representations.

However, UBL achieves much higher precision and better overall F1 scores, which are generally comparable to the best performing systems.

The comparison to the CCG induction techniques of ZC05 and ZC07 (Table 3) is particularly striking. These approaches used language-specific templates to propose new lexical items and also required as input a set of hand-engineered lexical entries to model phenomena such as quantification and determiners. However, the use of higher-order unification allows UBL to achieve comparable performance while automatically inducing these types of entries.

For a more qualitative evaluation, Table 4 shows a selection of lexical items learned with high weights for the lambda-calculus meaning representations. Nouns such as “state” or “estado” are consistently learned across languages with the category $S|NP$, which stands in for the more conventional N. The algorithm also learns language-specific constructions such as the Japanese case markers “no” and “wa”, which are treated as modifiers that do not add semantic content. Language-specific word order is also encoded, using the slash directions of the CCG categories. For example, “what” and “que” take their arguments to the right in the wh-initial English and Spanish. However, the Turkish wh-word “nelerdir” and the Japanese question marker “nan desu ka” are sentence final, and therefore take their arguments to the left. Learning regularities of this type allows UBL to generalize well to unseen data.

System	Variable Free			Lambda Calculus		
	Rec.	Pre.	F1	Rec.	Pre.	F1
Cross Validation Results						
KRISP	71.7	**93.3**	81.1	–	–	–
WASP	74.8	87.2	80.5	–	–	–
Lu08	81.5	89.3	**85.2**	–	–	–
λ-WASP	–	–	–	86.6	92.0	89.2
Independent Test Set						
ZC05	–	–	–	79.3	**96.3**	87.0
ZC07	–	–	–	86.1	91.6	88.8
UBL	81.4	89.4	**85.2**	85.0	94.1	**89.3**
UBL-s	**84.3**	85.2	84.7	**87.9**	88.5	88.2

Table 3: Performance on the Geo880 data set, with varied meaning representations.

There is less variation and complexity in the learned lexical items for the variable-free representation. The fact that the meaning representation is deeply nested influences the form of the induced grammar. For example, recall that the sentence “what states border texas” would be paired with the meaning $answer(state(borders(tex)))$. For this representation, lexical items such as:

$$\begin{aligned}&\text{what} \vdash S/NP : \lambda x.answer(x)\\&\text{states} \vdash NP/NP : \lambda x.state(x)\\&\text{border} \vdash NP/NP : \lambda x.borders(x)\\&\text{texas} \vdash NP : tex\end{aligned}$$

can be used to construct the desired output. In practice, UBL often learns entries with only a single slash, like those above, varying only in the direction, as required for the language. Even the more complex items, such as those for quantifiers, are consistently simpler than those induced from the lambda-calculus meaning representations. For example, one of the most complex entries learned in the experiments for English is the smallest $\vdash NP\backslash NP/(NP|NP){:}\lambda f \lambda x.smallest\_one(f(x))$.

There are also differences in the aggregate statistics of the learned lexicons. For example, the average length of a learned lexical item for the (lambda-calculus, variable-free) meaning representations is:

(1.21,1.08) for Turkish, (1.34,1.19) for English, (1.43,1.25) for Spanish and (1.63,1.42) for Japanese. For both meaning representations the model learns significantly more multiword lexical items for the somewhat analytic Japanese than the agglutinative Turkish. There are also variations in the average number of learned lexical items in the best parses during the final pass of training: 192 for Japanese, 206 for Spanish, 188 for English and 295 for Turkish. As compared to the other languages, the morpologically rich Turkish requires significantly more lexical variation to explain the data.

Finally, there are a number of cases where the UBL algorithm could be improved in future work. In cases where there are multiple allowable word orders, the UBL algorithm must learn individual entries for each possibility. For example, the following two categories are often learned with high weight for the Japanese word "chiisai":

$$NP/(S|NP)\backslash(NP|NP){:}\lambda f\lambda g.argmin(x, g(x), f(x))$$
$$NP|(S|NP)/(NP|NP){:}\lambda f\lambda g.argmin(x, g(x), f(x))$$

and are treated as distinct entries in the lexicon. Similarly, the approach presented here does not model morphology, and must repeatedly learn the correct categories for the Turkish words "nehri," "nehir," "nehirler," and "nehirlerin", all of which correspond to the logical form $\lambda x.river(x)$.

English
population of $\vdash NP/NP : \lambda x.population(x)$
smallest $\vdash NP/(S\vert NP) : \lambda f.arg\ min(y, f(y), size(y))$
what $\vdash S\vert NP/(S\vert NP) : \lambda f\lambda x.f(x)$
border $\vdash S\vert NP/NP : \lambda x\lambda y.next\_to(y, x)$
state $\vdash S\vert NP : \lambda x.state(x)$
most $\vdash NP/(S\vert NP)\backslash(S\vert NP)\backslash(S\vert NP\vert NP) :$ $\lambda f\lambda g\lambda h\lambda x.argmax(y, g(y), count(z, f(z, y) \wedge h(z)))$

Japanese
no $\vdash NP\vert NP/(NP\vert NP) : \lambda f\lambda x.f(x)$
shuu $\vdash S\vert NP : \lambda x.state(x)$
nan desu ka $\vdash S\backslash NP\backslash(NP\vert NP) : \lambda f\lambda x.f(x)$
wa $\vdash NP\vert NP\backslash(NP\vert NP) : \lambda f\lambda x.f(x)$
ikutsu $\vdash NP\vert(S\vert NP)\backslash(S\vert NP\vert(S\vert NP)) :$ $\lambda f\lambda g.count(x, f(g(x)))$
chiiki $\vdash NP\backslash NP{:}\lambda x.area(x)$

Turkish
nedir $\vdash S\backslash NP\backslash(NP\vert NP) : \lambda f\lambda x.f(x)$
sehir $\vdash S\vert NP : \lambda x.city(x)$
nufus yogunlugu $\vdash NP\vert NP : \lambda x.density(x)$
siniri$\vdash S\vert NP/NP : \lambda x\lambda y.next_to(y, x)$
kac tane $\vdash S\backslash NP/(S\vert NP\vert NP)\backslash(S\vert NP) :$ $\lambda f\lambda g\lambda x.count(y, f(y) \wedge g(y, x))$
ya siniri $\vdash S\vert NP\backslash NP : \lambda x\lambda y.next\_to(y, x)$

Spanish
en $\vdash S\vert NP/NP{:}\ \lambda x\lambda y.loc(y, x)$
que es la $\vdash S/NP/(NP\vert NP){:}\ \lambda f\lambda x.f(x)$
pequena $\vdash NP\backslash(S\vert NP)\backslash(NP\vert NP) :$ $\lambda g\lambda f.arg\ min(y, f(y), g(y))$
estado $\vdash S\vert NP : \lambda x.state(x)$
mas $\vdash S\backslash(S\vert NP)/(S\vert NP)\backslash(NP\vert NP\vert(S\vert NP)) :$ $\lambda f\lambda g\lambda h.argmax(x, h(x), f(g, x))$
mayores $\vdash S\vert NP\backslash(S\vert NP) : \lambda f\lambda x.f(x) \wedge major(x)$

Table 4: Example learned lexical items for each language on the Geo250 lambda-calculus data sets.

9 Conclusions and Future Work

This paper has presented a method for inducing probabilistic CCGs from sentences paired with logical forms. The approach uses higher-order unification to define the space of possible grammars in a language- and representation-independent manner, paired with an algorithm that learns a probabilistic parsing model. We evaluated the approach on four languages with two meaning representations each, achieving high accuracy across all scenarios.

For future work, we are interested in exploring the generality of the approach while extending it to new understanding problems. One potential limitation is in the constraints we introduced to ensure the tractability of the higher-order unification procedure. These restrictions will not allow the approach to induce lexical items that would be used with, among other things, many of the type-raised combinators commonly employed in CCG grammars. We are also interested in developing similar grammar induction techniques for context-dependent understanding problems, such as the one considered by Zettlemoyer & Collins (2009). Such an approach would complement ideas for using high-order unification to model a wider range of language phenomena, such as VP ellipsis (Dalrymple et al., 1991).

Acknowledgements

We thank the reviewers for useful feedback. This work was supported by the EU under IST Cognitive Systems grant IP FP6-2004-IST-4-27657 "Paco-Plus" and ERC Advanced Fellowship 249520 "GRAMPLUS" to Steedman. Kwiatkowski was supported by an EPRSC studentship. Zettlemoyer was supported by a US NSF International Research Fellowship.

References

Bos, J., Clark, S., Steedman, M., Curran, J. R., & Hockenmaier, J. (2004). Wide-coverage semantic representations from a CCG parser. In *Proceedings of the International Conference on Computational Linguistics*.

Buszkowski, W. & Penn, G. (1990). Categorial grammars determined from linguistic data by unification. *Studia Logica*, *49*, 431–454.

Carpenter, B. (1997). *Type-Logical Semantics*. The MIT Press.

Clark, S. & Curran, J. R. (2003). Log-linear models for wide-coverage CCG parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Clark, S. & Curran, J. R. (2007). Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, *33*(4), 493–552.

Dalrymple, M., Shieber, S., & Pereira, F. (1991). Ellipsis and higher-order unification. *Linguistics and Philosophy*, *14*, 399–452.

Ge, R. & Mooney, R. J. (2006). Discriminative reranking for semantic parsing. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*.

Huet, G. (1975). A unification algorithm for typed λ-calculus. *Theoretical Computer Science*, *1*, 27–57.

Huet, G. P. (1973). The undecidability of unification in third order logic. *Information and Control*, *22*(3), 257–267.

Kate, R. J. & Mooney, R. J. (2006). Using string-kernels for learning semantic parsers. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*.

Kate, R. J., Wong, Y. W., & Mooney, R. J. (2005). Learning to transform natural to formal languages. In *Proceedings of the National Conference on Artificial Intelligence*.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Lu, W., Ng, H. T., Lee, W. S., & Zettlemoyer, L. S. (2008). A generative model for parsing natural language to meaning representations. In *Proceedings of The Conference on Empirical Methods in Natural Language Processing*.

Miller, S., Stallard, D., Bobrow, R. J., & Schwartz, R. L. (1996). A fully statistical approach to natural language interfaces. In *Proc. of the Association for Computational Linguistics*.

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

Steedman, M. (1996). *Surface Structure and Interpretation*. The MIT Press.

Steedman, M. (2000). *The Syntactic Process*. The MIT Press.

Thompson, C. A. & Mooney, R. J. (2002). Acquiring word-meaning mappings for natural language interfaces. *Journal of Artificial Intelligence Research*, *18*.

Villavicencio, A. (2002). The acquisition of a unification-based generalised categorial grammar. Ph.D. thesis, University of Cambridge.

Wong, Y. W. & Mooney, R. (2006). Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL*.

Wong, Y. W. & Mooney, R. (2007). Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the Association for Computational Linguistics*.

Zelle, J. M. & Mooney, R. J. (1996). Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*.

Zettlemoyer, L. S. & Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Zettlemoyer, L. S. & Collins, M. (2007). Online learning of relaxed CCG grammars for parsing to logical form. In *Proc. of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Zettlemoyer, L. S. & Collins, M. (2009). Learning context-dependent mappings from sentences to logical form. In *Proceedings of The Joint Conference of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing*.

Using Universal Linguistic Knowledge to Guide Grammar Induction

Tahira Naseem, Harr Chen, Regina Barzilay
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{tahira, harr, regina} @csail.mit.edu

Mark Johnson
Department of Computing
Macquarie University
mark.johnson@mq.edu.au

Abstract

We present an approach to grammar induction that utilizes syntactic universals to improve dependency parsing across a range of languages. Our method uses a single set of manually-specified language-independent rules that identify syntactic dependencies between pairs of syntactic categories that commonly occur across languages. During inference of the probabilistic model, we use posterior expectation constraints to require that a minimum proportion of the dependencies we infer be instances of these rules. We also automatically refine the syntactic categories given in our coarsely tagged input. Across six languages our approach outperforms state-of-the-art unsupervised methods by a significant margin.[1]

1 Introduction

Despite surface differences, human languages exhibit striking similarities in many fundamental aspects of syntactic structure. These structural correspondences, referred to as *syntactic universals*, have been extensively studied in linguistics (Baker, 2001; Carnie, 2002; White, 2003; Newmeyer, 2005) and underlie many approaches in multilingual parsing. In fact, much recent work has demonstrated that learning cross-lingual correspondences from corpus data greatly reduces the ambiguity inherent in syntactic analysis (Kuhn, 2004; Burkett and Klein, 2008; Cohen and Smith, 2009a; Snyder et al., 2009; Berg-Kirkpatrick and Klein, 2010).

Root → Auxiliary	Noun → Adjective
Root → Verb	Noun → Article
Verb → Noun	Noun → Noun
Verb → Pronoun	Noun → Numeral
Verb → Adverb	Preposition → Noun
Verb → Verb	Adjective → Adverb
Auxiliary → Verb	

Table 1: The manually-specified universal dependency rules used in our experiments. These rules specify head-dependent relationships between coarse (i.e., unsplit) syntactic categories. An explanation of the ruleset is provided in Section 5.

In this paper, we present an alternative grammar induction approach that exploits these structural correspondences by declaratively encoding a small set of universal dependency rules. As input to the model, we assume a corpus annotated with coarse syntactic categories (i.e., high-level part-of-speech tags) and a set of universal rules defined over these categories, such as those in Table 1. These rules incorporate the definitional properties of syntactic categories in terms of their interdependencies and thus are universal across languages. They can potentially help disambiguate structural ambiguities that are difficult to learn from data alone — for example, our rules prefer analyses in which verbs are dependents of auxiliaries, even though analyzing auxiliaries as dependents of verbs is also consistent with the data. Leveraging these universal rules has the potential to improve parsing performance for a large number of human languages; this is particularly relevant to the processing of low-resource

[1] The source code for the work presented in this paper is available at http://groups.csail.mit.edu/rbg/code/dependency/

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1234–1244,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

languages. Furthermore, these universal rules are compact and well-understood, making them easy to manually construct.

In addition to these universal dependencies, each specific language typically possesses its own idiosyncratic set of dependencies. We address this challenge by requiring the universal constraints to only hold in expectation rather than absolutely, i.e., we permit a certain number of violations of the constraints.

We formulate a generative Bayesian model that explains the observed data while accounting for declarative linguistic rules during inference. These rules are used as expectation constraints on the posterior distribution over dependency structures. This approach is based on the posterior regularization technique (Graça et al., 2009), which we apply to a variational inference algorithm for our parsing model. Our model can also optionally refine common high-level syntactic categories into per-language categories by inducing a clustering of words using Dirichlet Processes (Ferguson, 1973). Since the universals guide induction toward linguistically plausible structures, automatic refinement becomes feasible even in the absence of manually annotated syntactic trees.

We test the effectiveness of our grammar induction model on six Indo-European languages from three language groups: English, Danish, Portuguese, Slovene, Spanish, and Swedish. Though these languages share a high-level Indo-European ancestry, they cover a diverse range of syntactic phenomenon. Our results demonstrate that universal rules greatly improve the accuracy of dependency parsing across all of these languages, outperforming current state-of-the-art unsupervised grammar induction methods (Headden III et al., 2009; Berg-Kirkpatrick and Klein, 2010).

2 Related Work

Learning with Linguistic Constraints Our work is situated within a broader class of unsupervised approaches that employ declarative knowledge to improve learning of linguistic structure (Haghighi and Klein, 2006; Chang et al., 2007; Graça et al., 2007; Cohen and Smith, 2009b; Druck et al., 2009; Liang et al., 2009a). The way we apply constraints is closest to the latter two approaches of posterior regularization and generalized expectation criteria.

In the posterior regularization framework, constraints are expressed in the form of expectations on posteriors (Graça et al., 2007; Ganchev et al., 2009; Graça et al., 2009; Ganchev et al., 2010). This design enables the model to reflect constraints that are difficult to encode via the model structure or as priors on its parameters. In their approach, parameters are estimated using a modified EM algorithm, where the E-step minimizes the KL-divergence between the model posterior and the set of distributions that satisfies the constraints. Our approach also expresses constraints as expectations on the posterior; we utilize the machinery of their framework within a variational inference algorithm with a mean field approximation.

Generalized expectation criteria, another technique for declaratively specifying expectation constraints, has previously been successfully applied to the task of dependency parsing (Druck et al., 2009). This objective expresses constraints in the form of preferences over model expectations. The objective is penalized by the square distance between model expectations and the prespecified values of the expectation. This approach yields significant gains compared to a fully unsupervised counterpart. The constraints they studied are corpus- and language-specific. Our work demonstrates that a small set of language-independent universals can also serve as effective constraints. Furthermore, we find that our method outperforms the generalized expectation approach using corpus-specific constraints.

Learning to Refine Syntactic Categories Recent research has demonstrated the usefulness of automatically refining the granularity of syntactic categories. While most of the existing approaches are implemented in the supervised setting (Finkel et al., 2007; Petrov and Klein, 2007), Liang et al. (2007) propose a non-parametric Bayesian model that learns the granularity of PCFG categories in an unsupervised fashion. For each non-terminal grammar symbol, the model posits a Hierarchical Dirichlet Process over its refinements (subsymbols) to automatically learn the granularity of syntactic categories. As with their work, we also use non-parametric priors for category refinement and employ variational methods for inference. However, our goal is to apply category refinement to dependency parsing, rather than to PCFGs, requiring a substantially different model formulation. While Liang et al. (2007) demonstrated empirical gains on a synthetic corpus, our experiments focus on unsupervised category refinement on real language data.

Universal Rules in NLP Despite the recent surge of interest in multilingual learning (Kuhn, 2004; Cohen and Smith, 2009a; Snyder et al., 2009; Berg-Kirkpatrick and Klein, 2010), there is surprisingly little computational work on linguistic universals. On the acquisition side, Daumé III and Campbell (2007) proposed a computational technique for discovering universal implications in typological features. More closely related to our work is the position paper by Bender (2009), which advocates the use of manually-encoded cross-lingual generalizations for the development of NLP systems. She argues that a system employing such knowledge could be easily adapted to a particular language by specializing this high level knowledge based on the typological features of the language. We also argue that cross-language universals are beneficial for automatic language processing; however, our focus is on learning language-specific adaptations of these rules from data.

3 Model

The central hypothesis of this work is that unsupervised dependency grammar induction can be improved using universal linguistic knowledge. Toward this end our approach is comprised of two components: a probabilistic model that explains how sentences are generated from latent dependency structures and a technique for incorporating declarative rules into the inference process.

We first describe the generative story in this section before turning to how constraints are applied during inference in Section 4. Our model takes as input (i.e., as observed) a set of sentences where each word is annotated with a coarse part-of-speech tag. Table 2 provides a detailed technical description of our model's generative process, and Figure 1 presents a model diagram.

For each observed coarse symbol s:

1. Draw top-level infinite multinomial over subsymbols $\beta_s \sim \mathrm{GEM}(\gamma)$.
2. For each subsymbol z of symbol s:
   (a) Draw word emission multinomial $\phi_{sz} \sim \mathrm{Dir}(\phi_0)$.
   (b) For each context value c:
      i. Draw child symbol generation multinomial $\theta_{szc} \sim \mathrm{Dir}(\theta_0)$.
      ii. For each child symbol s':
         A. Draw second-level infinite multinomial over subsymbols $\pi_{s'szc} \sim \mathrm{DP}(\alpha, \beta_{s'})$.

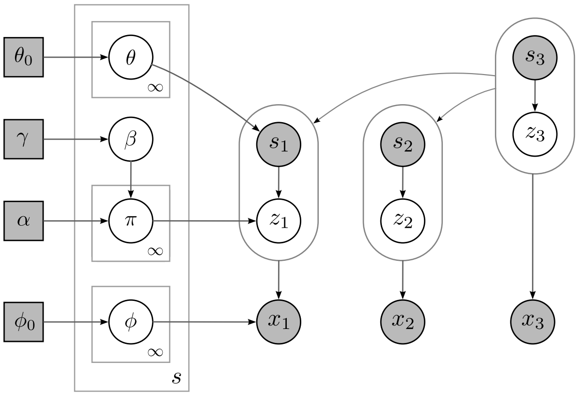
For each tree node i generated in context c by parent symbol s' and parent subsymbol z':

1. Draw coarse symbol $s_i \sim \mathrm{Mult}(\theta_{s'z'})$.
2. Draw subsymbol $z_i \sim \mathrm{Mult}(\pi_{s_i s' z' c})$.
3. Draw word $x_i \sim \mathrm{Mult}(\phi_{s_i z_i})$.

Table 2: The generative process for model parameters and parses. In the above GEM, DP, Dir, and Mult refer respectively to the stick breaking distribution, Dirichlet process, Dirichlet distribution, and multinomial distribution.

Generating Symbols and Words We describe how a single node of the tree is generated before discussing how the entire tree structure is formed. Each node of the dependency tree is comprised of three random variables: an observed coarse symbol s, a hidden refined subsymbol z, and an observed word x. In the following let the parent of the current node have symbol s' and subsymbol z'; the root node is generated from separate root-specific distributions. Subsymbol refinement is an optional component of the full model and can be omitted by deterministically equating s and z. As we explain at the end of this section, without this aspect the generative story closely resembles the classic dependency model with valence (DMV) of Klein and Manning (2004).

First we draw symbol s from a finite multinomial



s	-	coarse symbol (observed)
z	-	refined subsymbol
x	-	word (observed)
θ_{szc}	-	distr over child coarse symbols for each parent s and z and context c
β_s	-	top-level distr over subsymbols for s
$\pi_{ss'z'c}$	-	distr over subsymbols for each s, parent s' and z', and context c
ϕ_{sz}	-	distr over words for s and z

Figure 1: Graphical representation of the model and a summary of the notation. There is a copy of the outer plate for each distinct symbol in the observed coarse tags. Here, node 3 is shown to be the parent of nodes 1 and 2. Shaded variables are observed, square variables are hyperparameters. The elongated oval around s and z represents the two variables jointly. For clarity the diagram omits some arrows from θ to each s, π to each z, and ϕ to each x.

distribution with parameters $\theta_{s'z'c}$. As the indices indicate, we have one such set of multinomial parameters for every combination of parent symbol s' and subsymbol z' along with a *context* c. Here the context of the current node can take one of six values corresponding to every combination of direction (left or right) and valence (first, second, or third or higher child) with respect to its parent. The prior (base distribution) for each $\theta_{s'z'c}$ is a symmetric Dirichlet with hyperparameter θ_0.

Next we draw the refined syntactic category subsymbol z from an infinite multinomial with parameters $\pi_{ss'z'c}$. Here the selection of π is indexed by the current node's coarse symbol s, the symbol s' and subsymbol z' of the parent node, and the context c of the current node.

For each unique coarse symbol s we tie together the distributions $\pi_{ss'z'c}$ for all possible parent and context combinations (i.e., s', z', and c) using a Hierarchical Dirichlet Process (HDP). Specifically, for a single s, each distribution $\pi_{ss'z'c}$ over subsymbols is drawn from a DP with concentration parameter α and base distribution β_s over subsymbols. This base distribution β_s is itself drawn from a GEM prior with concentration parameter γ. By formulating the generation of z as an HDP, we can share parameters for a single coarse symbol's subsymbol distribution while allowing for individual variability based on node parent and context. Note that parameters are not shared across different coarse symbols, preserving the distinctions expressed via the coarse tag annotations.

Finally, we generate the word x from a finite multinomial with parameters ϕ_{sz}, where s and z are the symbol and subsymbol of the current node. The ϕ distributions are drawn from a symmetric Dirichlet prior.

Generating the Tree Structure We now consider how the structure of the tree arises. We follow an approach similar to the widely-referenced DMV model (Klein and Manning, 2004), which forms the basis of the current state-of-the-art unsupervised grammar induction model (Headden III et al., 2009). After a node is drawn we generate children on each side until we produce a designated STOP symbol. We encode more detailed valence information than Klein and Manning (2004) and condition child generation on parent valence. Specifically, after drawing a node we first decide whether to proceed to generate a child or to stop conditioned on the parent symbol and subsymbol and the current context (direction and valence). If we decide to generate a child we follow the previously described process for constructing a node. We can combine the stopping decision with the generation of the child symbol by including a distinguished STOP symbol as a possible outcome in distribution θ.

No-Split Model Variant In the absence of subsymbol refinement (i.e., when subsymbol z is set to be identical to coarse symbol s), our model simplifies in some respects. In particular, the HDP generation of z is obviated and word x is drawn from a word distribution ϕ_s indexed solely by coarse symbol s. The resulting simplified model closely resembles DMV (Klein and Manning, 2004), except that it 1) explicitly generate words x rather than only part-of-speech tags s, 2) encodes richer context and valence information, and 3) imposes a Dirichlet prior on the symbol distribution θ.

4 Inference with Constraints

We now describe how to augment our generative model of dependency structure with constraints derived from linguistic knowledge. Incorporating arbitrary linguistic rules directly in the generative story is challenging as it requires careful tuning of either the model structure or priors for each constraint. Instead, following the approach of Graça et al. (2007), we constrain the posterior to satisfy the rules in expectation during inference. This effectively biases the inference toward linguistically plausible settings.

In standard variational inference, an intractable true posterior is approximated by a distribution from a tractable set (Bishop, 2006). This tractable set typically makes stronger independence assumptions between model parameters than the model itself. To incorporate the constraints, we further restrict the set to only include distributions that satisfy the specified expectation constraints over hidden variables.

In general, for some given model, let θ denote the entire set of model parameters and z and x denote the hidden structure and observations respectively. We are interested in estimating the posterior $p(\theta, z \mid x)$. Variational inference transforms this problem into an optimization problem where we try to find a distribution $q(\theta, z)$ from a restricted set $\mathcal{Q}$ that minimizes the KL-divergence between $q(\theta, z)$ and $p(\theta, z \mid x)$:

$$\begin{aligned}&\mathrm{KL}(q(\theta, z) \parallel p(\theta, z \mid x)) \\ &\quad = \int q(\theta, z) \log \frac{q(\theta, z)}{p(\theta, z, x)} d\theta dz + \log p(x).\end{aligned}$$

Rearranging the above yields:

$$\log p(x) = \mathrm{KL}(q(\theta, z) \parallel p(\theta, z \mid x)) + \mathcal{F},$$

where $\mathcal{F}$ is defined as

$$\mathcal{F} \equiv \int q(\theta, z) \log \frac{p(\theta, z, x)}{q(\theta, z)} d\theta dz. \quad (1)$$

Thus $\mathcal{F}$ is a lower bound on likelihood. Maximizing this lower bound is equivalent to minimizing the KL-divergence between $p(\theta, z \mid x)$ and $q(\theta, z)$. To make this maximization tractable we make a mean field assumption that q belongs to a set $\mathcal{Q}$ of distributions that factorize as follows:

$$q(\theta, z) = q(\theta)q(z).$$

We further constrain q to be from the subset of $\mathcal{Q}$ that satisfies the expectation constraint $E_q[f(z)] \leq b$ where f is a deterministically computable function of the hidden structures. In our model, for example, f counts the dependency edges that are an instance of one of the declaratively specified dependency rules, while b is the proportion of the total dependencies that we expect should fulfill this constraint.[2]

With the mean field factorization and the expectation constraints in place, solving the maximization of $\mathcal{F}$ in (1) separately for each factor yields the following updates:

$$q(\theta) = \underset{q(\theta)}{\operatorname{argmin}} \ \mathrm{KL}\left(q(\theta) \parallel q'(\theta)\right), \quad (2)$$

$$q(z) = \underset{q(z)}{\operatorname{argmin}} \ \mathrm{KL}\left(q(z) \parallel q'(z)\right)$$

$$s.t. \quad E_{q(z)}[f(z)] \leq b, \quad (3)$$

where

$$q'(\theta) \propto \exp E_{q(z)}[\log p(\theta, z, x)], \quad (4)$$

$$q'(z) \propto \exp E_{q(\theta)}[\log p(\theta, z, x)]. \quad (5)$$

We can solve (2) by setting $q(\theta)$ to $q'(\theta)$ — since $q(z)$ is held fixed while updating $q(\theta)$, the expectation function of the constraint remains constant during this update. As shown by Graça et al. (2007), the update in (3) is a constrained optimization problem and can be solved by performing gradient search on its dual:

$$\underset{\lambda}{\operatorname{argmin}} \ \lambda^\top b + \log \sum_z q'(z) \exp(-\lambda^\top f(z)) \quad (6)$$

For a fixed value of λ the optimal $q(z) \propto q'(z) \exp(-\lambda^\top f(z))$. By updating $q(\theta)$ and $q(z)$ as in (2) and (3) we are effectively maximizing the lower bound $\mathcal{F}$.

[2] Constraints of the form $E_q[f(z)] \geq b$ are easily imposed by negating $f(z)$ and b.

4.1 Variational Updates

We now derive the specific variational updates for our dependency induction model. First we assume the following mean-field factorization of our variational distribution:

$$q(\beta, \theta, \pi, \phi, z) = q(z) \cdot \prod_{s'} q(\beta_{s'}) \cdot \prod_{z'=1}^{T} q(\phi_{s'z'}) \cdot \prod_{c} q(\theta_{s'z'c}) \cdot \prod_{s} q(\pi_{ss'z'c}), \tag{7}$$

where s' varies over the set of unique symbols in the observed tags, z' denotes subsymbols for each symbol, c varies over context values comprising a pair of direction (left or right) and valence (first, second, or third or higher) values, and s corresponds to child symbols.

We restrict $q(\theta_{s'z'c})$ and $q(\phi_{s'z'})$ to be Dirichlet distributions and $q(z)$ to be multinomial. As with prior work (Liang et al., 2009b), we assume a degenerate $q(\beta) \equiv \delta_{\beta^*}(\beta)$ for tractability reasons, i.e., all mass is concentrated on some single β^*. We also assume that the top level stick-breaking distribution is truncated at T, i.e., $q(\beta)$ assigns zero probability to integers greater than T. Because of the truncation of β, we can approximate $q(\pi_{ss'z'c})$ with an asymmetric finite dimensional Dirichlet.

The factors are updated one at a time holding all other factors fixed. The variational update for $q(\pi)$ is given by:

$$q(\pi_{ss'z'c}) = \text{Dir}\left(\pi_{ss'z'c}; \alpha\beta + E_{q(z)}[C_{ss'z'c}(z)]\right),$$

where term $E_{q(z)}[C_{ss'z'c}(z)]$ is the expected count w.r.t. $q(z)$ of child symbol s and subsymbol z in context c when generated by parent symbol s' and subsymbol z'.

Similarly, the updates for $q(\theta)$ and $q(\phi)$ are given by:

$$q(\theta_{s'z'c}) = \text{Dir}\left(\theta_{s'z'c}; \theta_0 + E_{q(z)}[C_{s'z'c}(s)]\right),$$
$$q(\phi_{s'z'}) = \text{Dir}\left(\phi_{s'z'}; \phi_0 + E_{q(z)}[C_{s'z'}(x)]\right),$$

where $C_{s'z'c}(s)$ is the count of child symbol s being generated by the parent symbol s' and subsymbol z' in context c and $C_{s'z'x}$ is the count of word x being generated by symbol s' and subsymbol z'.

The only factor affected by the expectation constraints is $q(z)$. Recall from the previous section that the update for $q(z)$ is performed via gradient search on the dual of a constrained minimization problem of the form:

$$q(z) = \operatorname*{argmin}_{q(z)} \text{KL}(q(z) \parallel q'(z)).$$

Thus we first compute the update for $q'(z)$:

$$q'(z) \propto \prod_{n=1}^{N} \prod_{j=1}^{len(n)} \big(\exp E_{q(\phi)}[\log \phi_{s_{nj} z_{nj}}(x_{nj})] \times \exp E_{q(\theta)}[\log \theta_{s_{h(nj)} z_{h(nj)} c_{nj}}(s_{nj})] \times \exp E_{q(\pi)}[\log \pi_{s_{nj} s_{h(nj)} z_{h(nj)} c_{nj}}(z_{nj})]\big),$$

where N is the total number of sentences, $len(n)$ is the length of sentence n, and index $h(nj)$ refers to the head of the jth node of sentence n. Given this $q'(z)$ a gradient search is performed using (6) to find the optimal λ and thus the primal solution for updating $q(z)$.

Finally, we update the degenerate factor $q(\beta_s)$ with the projected gradient search algorithm used by Liang et al. (2009b).

5 Linguistic Constraints

Universal Dependency Rules We compile a set of 13 universal dependency rules consistent with various linguistic accounts (Carnie, 2002; Newmeyer, 2005), shown in Table 1. These rules are defined over coarse part-of-speech tags: Noun, Verb, Adjective, Adverb, Pronoun, Article, Auxiliary, Preposition, Numeral and Conjunction. Each rule specifies a part-of-speech for the head and argument but does not provide ordering information.

We require that a minimum proportion of the posterior dependencies be instances of these rules in expectation. In contrast to prior work on rule-driven dependency induction (Druck et al., 2009), where each rule has a separately specified expectation, we only set a single minimum expectation for the proportion of all dependencies that must match one of the rules. This setup is more relevant for learning with universals since individual rule frequencies vary greatly between languages.

1. Identify non-recursive NPs:
   - All nouns, pronouns and possessive marker are part of an NP.
   - All adjectives, conjunctions and determiners immediately preceding an NP are part of the NP.
2. The first verb or modal in the sentence is the headword.
3. All words in an NP are headed by the last word in the NP.
4. The last word in an NP is headed by the word immediately before the NP if it is a preposition, otherwise it is headed by the headword of the sentence if the NP is before the headword, else it is headed by the word preceding the NP.
5. For the first word set its head to be the headword of the sentence. For each other word set its headword to be the previous word.

Table 3: English-specific dependency rules.

English-specific Dependency Rules For English, we also consider a small set of hand-crafted dependency rules designed by Michael Collins[3] for deterministic parsing, shown in Table 3. Unlike the universals from Table 1, these rules alone are enough to construct a full dependency tree. Thus they allow us to judge whether the model is able to improve upon a human-engineered deterministic parser. Moreover, with this dataset we can assess the additional benefit of using rules tailored to an individual language as opposed to universal rules.

6 Experimental Setup

Datasets and Evaluation We test the effectiveness of our grammar induction approach on English, Danish, Portuguese, Slovene, Spanish, and Swedish. For English we use the Penn Treebank (Marcus et al., 1993), transformed from CFG parses into dependencies with the Collins head finding rules (Collins, 1999); for the other languages we use data from the 2006 CoNLL-X Shared Task (Buchholz and Marsi, 2006). Each dataset provides manually annotated part-of-speech tags that are used for both training and testing. For comparison purposes with previous work, we limit the cross-lingual experiments to sentences of length 10 or less (not counting punctuation). For English, we also explore sentences of length up to 20.

The final output metric is directed dependency accuracy. This is computed based on the Viterbi parses produced using the final unnormalized variational distribution $q(z)$ over dependency structures.

Hyperparameters and Training Regimes Unless otherwise stated, in experiments with rule-based constraints the expected proportion of dependencies that must satisfy those constraints is set to 0.8. This threshold value was chosen based on minimal tuning on a single language and ruleset (English with universal rules) and carried over to each other experimental condition. A more detailed discussion of the threshold's empirical impact is presented in Section 7.1.

Variational approximations to the HDP are truncated at 10. All hyperparameter values are fixed to 1 except α which is fixed to 10.

We also conduct a set of *No-Split* experiments to evaluate the importance of syntactic refinement; in these experiments each coarse symbol corresponds to only one refined symbol. This is easily effected during inference by setting the HDP variational approximation truncation level to one.

For each experiment we run 50 iterations of variational updates; for each iteration we perform five steps of gradient search to compute the update for the variational distribution $q(z)$ over dependency structures.

7 Results

In the following section we present our primary cross-lingual results using universal rules (Section 7.1) before performing a more in-depth analysis of model properties such as sensitivity to ruleset selection and inference stability (Section 7.2).

[3] Personal communication.

	DMV	PGI	No-Split	HDP-DEP
English	47.1	62.3	71.5	**71.9** (0.3)
Danish	33.5	41.6	48.8	**51.9** (1.6)
Portuguese	38.5	63.0	54.0	**71.5** (0.5)
Slovene	38.5	48.4	50.6	**50.9** (5.5)
Spanish	28.0	58.4	64.8	**67.2** (0.4)
Swedish	45.3	58.3	**63.3**	62.1 (0.5)

Table 4: Directed dependency accuracy using our model with universal dependency rules (No-Split and HDP-DEP), compared to DMV (Klein and Manning, 2004) and PGI (Berg-Kirkpatrick and Klein, 2010). The DMV results are taken from Berg-Kirkpatrick and Klein (2010). Bold numbers indicate the best result for each language. For the full model, the standard deviation in performance over five runs is indicated in parentheses.

7.1 Main Cross-Lingual Results

Table 4 shows the performance of both our full model (HDP-DEP) and its No-Split version using universal dependency rules across six languages. We also provide the performance of two baselines — the dependency model with valence (DMV) (Klein and Manning, 2004) and the phylogenetic grammar induction (PGI) model (Berg-Kirkpatrick and Klein, 2010).

HDP-DEP outperforms both DMV and PGI across all six languages. Against DMV we achieve an average absolute improvement of 24.1%. This improvement is expected given that DMV does not have access to the additional information provided through the universal rules. PGI is more relevant as a point of comparison, since it is able to leverage multilingual data to learn information similar to what we have declaratively specified using universal rules. Specifically, PGI reduces induction ambiguity by connecting language-specific parameters via phylogenetic priors. We find, however, that we outperform PGI by an average margin of 7.2%, demonstrating the benefits of explicit rule specification.

An additional point of comparison is the lexicalized unsupervised parser of Headden III et al. (2009), which yields the current state-of-the-art unsupervised accuracy on English at 68.8%. Our method also outperforms this approach, without employing lexicalization and sophisticated smoothing as they do. This result suggests that combining the complementary strengths of their approach and ours can yield further performance improvements.

English			
Rule Excluded	Acc	Loss	Gold Freq
Preposition → Noun	61.0	10.9	5.1
Verb → Noun	61.4	10.5	14.8
Noun → Noun	64.4	7.5	10.7
Noun → Article	64.7	7.2	8.5

Spanish			
Rule Excluded	Acc	Loss	Gold Freq
Preposition → Noun	53.4	13.8	8.2
Verb → Noun	61.9	5.4	12.9
Noun → Noun	62.6	4.7	2.0
Root → Verb	65.4	1.8	12.3

Table 5: Ablation experiment results for universal dependency rules on English and Spanish. For each rule, we evaluate the model using the ruleset excluding that rule, and list the most significant rules for each language. The second last column is the absolute loss in performance compared to the setting where all rules are available. The last column shows the percentage of the gold dependencies that satisfy the rule.

Table 4 also shows the *No-Split* results where syntactic categories are not refined. We find that such refinement usually proves to be beneficial, yielding an average performance gain of 3.7%. However, we note that the impact of incorporating splitting varies significantly across languages. Further understanding of this connection is an area of future research.

Finally, we note that our model exhibits low variance for most languages. This result attests to how the expectation constraints consistently guide inference toward high-accuracy areas of the search space.

Ablation Analysis Our next experiment seeks to understand the relative importance of the various universal rules from Table 1. We study how accuracy is affected when each of the rules is removed one at a time for English and Spanish. Table 5 lists the rules with the greatest impact on performance when removed. We note the high overlap between the most significant rules for English and Spanish.

We also observe that the relationship between a rule's frequency and its importance for high accuracy is not straightforward. For example, the "Preposition → Noun" rule, whose removal degrades accuracy the most for both English and Span-

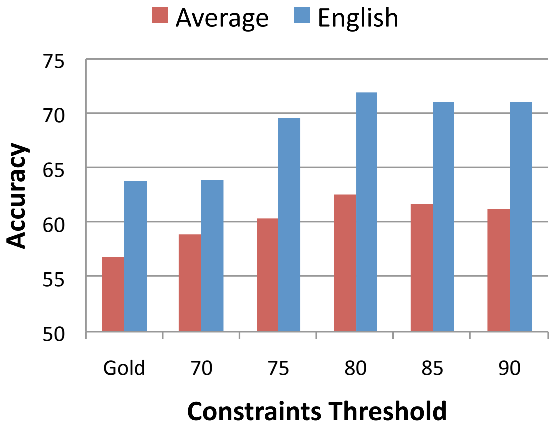


Figure 2: Accuracy of our model with different threshold settings, on English only and averaged over all languages. "Gold" refers to the setting where each language's threshold is set independently to the proportion of gold dependencies satisfying the rules — for English this proportion is 70%, while the average proportion across languages is 63%.

ish, is not the most frequent rule in either language. This result suggests that some rules are harder to learn than others regardless of their frequency, so their presence in the specified ruleset yields stronger performance gains.

Varying the Constraint Threshold In our main experiments we require that at least 80% of the expected dependencies satisfy the rule constraints. We arrived at this threshold by tuning on the basis of English only. As shown in Figure 2, for English a broad band of threshold values from 75% to 90% yields results within 2.5% of each other, with a slight peak at 80%.

To further study the sensitivity of our method to how the threshold is set, we perform *post hoc* experiments with other threshold values on each of the other languages. As Figure 2 also shows, on average a value of 80% is optimal across languages, though again accuracy is stable within 2.5% between thresholds of 75% to 90%. These results demonstrate that a single threshold is broadly applicable across languages.

Interestingly, setting the threshold value independently for each language to its "true" proportion based on the gold dependencies (denoted as the "Gold" case in Figure 2) does not achieve optimal performance. Thus, knowledge of the true language-specific rule proportions is not necessary for high accuracy.

		Length	
		≤ 10	≤ 20
Universal Dependency Rules			
1	HDP-DEP	71.9	50.4
No Rules (Random Init)			
2	HDP-DEP	24.9	24.4
3	Headden III et al. (2009)	68.8	-
English-Specific Parsing Rules			
4	Deterministic (rules only)	70.0	62.6
5	HDP-DEP	**73.8**	**66.1**
Druck et al. (2009) Rules			
6	Druck et al. (2009)	61.3	-
7	HDP-DEP	64.9	42.2

Table 6: Directed accuracy of our model (HDP-DEP) on sentences of length 10 or less and 20 or less from WSJ with different rulesets and with no rules, along with various baselines from the literature. Entries in this table are numbered for ease of reference in the text.

7.2 Analysis of Model Properties

We perform a set of additional experiments on English to gain further insight into HDP-DEP's behavior. Our choice of language is motivated by the fact that a wide range of prior parsing algorithms were developed for and tested exclusively on English. The experiments below demonstrate that 1) universal rules alone are powerful, but language- and dataset-tailored rules can further improve performance; 2) our model learns jointly from the rules and data, outperforming a rules-only deterministic parser; 3) the way we incorporate posterior constraints outperforms the generalized expectation constraint framework; and 4) our model exhibits low variance when seeded with different initializations. These results are summarized in Table 6 and discussed in detail below; line numbers refer to entries in Table 6. Each run of HDP-DEP below is with syntactic refinement enabled.

Impact of Rules Selection We compare the performance of HDP-DEP using the universal rules versus a set of rules designed for deterministically parsing the Penn Treebank (see Section 5 for details).

As lines 1 and 5 of Table 6 show, language-specific rules yield better performance. For sentences of length 10 or less, the difference between the two rulesets is a relatively small 1.9%; for longer sentences, however, the difference is a substantially larger 15.7%. This is likely because longer sentences tend to be more complex and thus exhibit more language-idiosyncratic dependencies. Such dependencies can be better captured by the refined language-specific rules.

We also test model performance when no linguistic rules are available, i.e., performing unconstrained variational inference. The model performs substantially worse (line 2), confirming that syntactic category refinement in a fully unsupervised setup is challenging.

Learning Beyond Provided Rules Since HDP-DEP is provided with linguistic rules, a legitimate question is whether it improves upon what the rules encode, especially when the rules are complete and language-specific. We can answer this question by comparing the performance of our model seeded with the English-specific rules against a deterministic parser that implements the same rules. Lines 4 and 5 of Table 6 demonstrate that the model outperforms a rules-only deterministic parser by 3.8% for sentences of length 10 or less and by 3.5% for sentences of length 20 or less.

Comparison with Alternative Semi-supervised Parser The dependency parser based on the generalized expectation criteria (Druck et al., 2009) is the closest to our reported work in terms of technique. To compare the two, we run HDP-DEP using the 20 rules given by Druck et al. (2009). Our model achieves an accuracy of 64.9% (line 7) compared to 61.3% (line 6) reported in their work. Note that we do not rely on rule-specific expectation information as they do, instead requiring only a single expectation constraint parameter.[4]

Model Stability It is commonly acknowledged in the literature that unsupervised grammar induction methods exhibit sensitivity to initialization. As in the previous section, we find that the presence of linguistic rules greatly reduces this sensitivity: for HDP-DEP, the standard deviation over five randomly initialized runs with the English-specific rules is 1.5%, compared to 4.5% for the parser developed by Headden III et al. (2009) and 8.0% for DMV (Klein and Manning, 2004).

8 Conclusions

In this paper we demonstrated that syntactic universals encoded as declarative constraints improve grammar induction. We formulated a generative model for dependency structure that models syntactic category refinement and biases inference to cohere with the provided constraints. Our experiments showed that encoding a compact, well-accepted set of language-independent constraints significantly improves accuracy on multiple languages compared to the current state-of-the-art in unsupervised parsing.

While our present work has yielded substantial gains over previous unsupervised methods, a large gap still remains between our method and fully supervised techniques. In future work we intend to study ways to bridge this gap by 1) incorporating more sophisticated linguistically-driven grammar rulesets to guide induction, 2) lexicalizing the model, and 3) combining our constraint-based approach with richer unsupervised models (e.g., Headden III et al. (2009)) to benefit from their complementary strengths.

Acknowledgments

The authors acknowledge the support of the NSF (CAREER grant IIS-0448168, grant IIS-0904684, and a Graduate Research Fellowship). We are especially grateful to Michael Collins for inspiring us toward this line of inquiry and providing deterministic rules for English parsing. Thanks to Taylor Berg-Kirkpatrick, Sabine Iatridou, Ramesh Sridharan, and members of the MIT NLP group for their suggestions and comments. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of the funding organizations.

[4] As explained in Section 5, having a single expectation parameter is motivated by our focus on parsing with universal rules.

References

Mark C. Baker. 2001. *The Atoms of Language: The Mind's Hidden Rules of Grammar*. Basic Books.

Emily M. Bender. 2009. Linguistically naïve != language independent: Why NLP needs linguistic typology. In *Proceedings of the EACL 2009 Workshop on the Interaction between Linguistics and Computational Linguistics: Virtuous, Vicious or Vacuous?*, pages 26–32.

Taylor Berg-Kirkpatrick and Dan Klein. 2010. Phylogenetic grammar induction. In *Proceedings of ACL*, pages 1288–1297.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL*, pages 149–164.

David Burkett and Dan Klein. 2008. Two languages are better than one (for syntactic parsing). In *Proceedings of EMNLP*, pages 877–886.

Andrew Carnie. 2002. *Syntax: A Generative Introduction (Introducing Linguistics)*. Blackwell Publishing.

Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *Proceedings of ACL*, pages 280–287.

Shay B. Cohen and Noah A. Smith. 2009a. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of NAACL/HLT*, pages 74–82.

Shay B. Cohen and Noah A. Smith. 2009b. Variational inference for grammar induction with prior knowledge. In *Proceedings of ACL/IJCNLP 2009 Conference Short Papers*, pages 1–4.

Michael Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania.

Hal Daumé III and Lyle Campbell. 2007. A bayesian model for discovering typological implications. In *Proceedings of ACL*, pages 65–72.

Gregory Druck, Gideon Mann, and Andrew McCallum. 2009. Semi-supervised learning of dependency parsers using generalized expectation criteria. In *Proceedings of ACL/IJCNLP*, pages 360–368.

Thomas S. Ferguson. 1973. A bayesian analysis of some nonparametric problems. *Annals of Statistics*, 1(2):209–230.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2007. The infinite tree. In *Proceedings of ACL*, pages 272–279.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of ACL/IJCNLP*, pages 369–377.

Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11:2001–2049.

João Graça, Kuzman Ganchev, Ben Taskar, and Fernando Pereira. 2009. Posterior vs. parameter sparsity in latent variable models. In *Advances in NIPS*, pages 664–672.

João Graça, Kuzman Ganchev, and Ben Taskar. 2007. Expectation maximization and posterior constraints. In *Advances in NIPS*, pages 569–576.

Aria Haghighi and Dan Klein. 2006. Prototype-driven grammar induction. In *Proceedings of ACL*, pages 881–888.

William P. Headden III, Mark Johnson, and David McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of NAACL/HLT*, pages 101–109.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of ACL*, pages 478–485.

Jonas Kuhn. 2004. Experiments in parallel-text based grammar induction. In *Proceedings of ACL*, pages 470–477.

Percy Liang, Slav Petrov, Michael Jordan, and Dan Klein. 2007. The infinite PCFG using hierarchical Dirichlet processes. In *Proceedings of EMNLP/CoNLL*, pages 688–697.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009a. Learning from measurements in exponential families. In *Proceedings of ICML*, pages 641–648.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009b. Probabilistic grammars and hierarchical Dirichlet processes. *The Handbook of Applied Bayesian Analysis*.

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.

Frederick J. Newmeyer. 2005. *Possible and Probable Languages: A Generative Perspective on Linguistic Typology*. Oxford University Press.

Slav Petrov and Dan Klein. 2007. Learning and inference for hierarchically split PCFGs. In *Proceeding of AAAI*, pages 1663–1666.

Benjamin Snyder, Tahira Naseem, and Regina Barzilay. 2009. Unsupervised multilingual grammar induction. In *Proceedings of ACL/IJCNLP*, pages 73–81.

Lydia White. 2003. *Second Language Acquisition and Universal Grammar*. Cambridge University Press.

What's with the Attitude? Identifying Sentences with Attitude in Online Discussions

Ahmed Hassan **Vahed Qazvinian** **Dragomir Radev**
University of Michigan Ann Arbor
Ann Arbor, Michigan, USA
hassanam,vahed,radev@umich.edu

Abstract

Mining sentiment from user generated content is a very important task in Natural Language Processing. An example of such content is threaded discussions which act as a very important tool for communication and collaboration in the Web. Threaded discussions include e-mails, e-mail lists, bulletin boards, newsgroups, and Internet forums. Most of the work on sentiment analysis has been centered around finding the sentiment toward products or topics. In this work, we present a method to identify the attitude of participants in an online discussion toward one another. This would enable us to build a signed network representation of participant interaction where every edge has a sign that indicates whether the interaction is positive or negative. This is different from most of the research on social networks that has focused almost exclusively on positive links. The method is experimentally tested using a manually labeled set of discussion posts. The results show that the proposed method is capable of identifying attitudinal sentences, and their signs, with high accuracy and that it outperforms several other baselines.

1 Introduction

Mining sentiment from text has a wide range of applications from mining product reviews on the Web (Morinaga et al., 2002; Turney and Littman, 2003) to analyzing political speeches (Thomas et al., 2006). Automatic methods for sentiment mining are very important because manual extraction of them is very costly, and inefficient. A new application of sentiment mining is to automatically identify attitudes between participants in an online discussion. An automatic tool to identify attitudes will enable us to build a signed network representation of participant interaction in which the interaction between two participants is represented using a positive or a negative edge. Even though using signed edges in social network studies is clearly important, most of the social networks research has focused only on positive links between entities. Some work has recently investigated signed networks (Leskovec et al., 2010; Kunegis et al., 2009), however this work was limited to a few number of datasets in which users were allowed to explicitly add negative, as well as positive, relations. This work will pave the way for research efforts to examine signed social networks in more detail. It will also allow us to study the relation between explicit relations and the text underlying those relation.

Although similar, identifying sentences that display an attitude in discussions is different from identifying opinionated sentences. A sentence in a discussion may bear opinions about a definite target (e.g., price of a camera) and yet have no attitude toward the other participants in the discussion. For instance, in the following discussion Alice's sentence has her opinion against something, yet no attitude toward the recipient of the sentence, Bob.

> Alice: "You know what, he turned out to be a great disappointment"
> Bob: "You are completely unqualified to judge this great person"

However, Bob shows strong attitude toward Alice. In this work, we look at ways to predict whether a sentence displays an attitude toward the text recipient. An attitude is the mental position of one participant with regard to another participant. it could be either positive or negative. We consider features which takes into account the entire structure of sentences at different levels or generalization. Those

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1245–1255,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

features include lexical items, part-of-speech tags, and dependency relations. We use all those patterns to build several pairs of models that represent sentences with and without attitude.

The rest of the paper is organized as follows. In Section 2 we review some of the related prior work on identifying polarized words and subjectivity analysis. We explain the problem definition and discuss our approach in Sections 3 & 4. Finally, in Sections 5 & 6 we introduce our dataset and discuss the experimental setup. Finally, we conclude in Section 7.

2 Related Work

Identifying the polarity of individual words is a well studied problem. In previous work, Hatzivassiloglou and McKeown (1997) propose a method to identify the polarity of adjectives. They use a manually labeled corpus to classify each conjunction of an adjective as "the same orientation" as the adjective or "different orientation". Their method can label simple in "simple and well-received" as the same orientation and simplistic in "simplistic but well-received" as the opposite orientation of well-received. Although the results look promising, the method would only be applicable to adjectives since noun conjunctions may collocate regardless of their semantic orientations (e.g., "rise and fall").

In other work, Turney and Littman (2003) use statistical measures to find the association between a given word and a set of positive/negative seed words. In order to get word co-occurrence statistics they use the "near" operator from a commercial search engine on a given word and a seed word.

In more recent work, Takamura et al. (2005) used the spin model to extract word semantic orientation. First, they construct a network of words using definitions, thesaurus, and co-occurrence statistics. In this network, each word is regarded as an electron, which has a spin and each spin has a direction taking one of two values: up or down. Then, they use the energy point of view to propose that neighboring electrons tend to have the same spin direction, and therefore neighboring words tend to have the same polarity orientations. Finally, they use the mean field method to find the optimal solution for electron spin directions.

Previous work has also used WordNet, a lexical database of English, to identify word polarity. Specifically, Hu and Liu (2004) use WordNet synonyms and antonyms to predict the polarity of any given word with unknown polarity. They label each word with the polarity of its synonyms and the opposite polarity of its antonyms. They continue in a bootstrapping manner to label all unlabeled instances. This work is very similar to (Kamps et al., 2004) in which a network of WordNet synonyms is used to find the shortest path between any given word, and the words "good" and "bad". Kim and Hovy (Kim and Hovy, 2004) used WordNet synonyms and antonyms to expand two lists of positive and negative seed words. Similarly, Andreevskaia and Bergler (2006) used WordNet to expand seed lists with fuzzy sentiment categories, in which words could be more central to one category than the other. Finally, Kanayama and Nasukawa (2006) used syntactic features and context coherency, defined as the tendency for same polarities to appear successively, to acquire polar atoms.

All the work mentioned above focus on the task of identifying the polarity of individual words. Our proposed work is identifying attitudes in sentences that appear in online discussions. Perhaps the most similar work to ours is the prior work on subjectivity analysis, which is to identify text that present opinions as opposed to objective text that present factual information (Wiebe, 2000). Prior work on subjectivity analysis mainly consists of two main categories: The first category is concerned with identifying the subjectivity of individual phrases and words regardless of the sentence and context they appear in (Wiebe, 2000; Hatzivassiloglou and Wiebe, 2000; Banea et al., 2008). In the second category, subjectivity of a phrase or word is analyzed within its context (Riloff and Wiebe, 2003; Yu and Hatzivassiloglou, 2003; Nasukawa and Yi, 2003; Popescu and Etzioni,). A good study of the applications of subjectivity analysis from review mining to email classification is given in (Wiebe, 2000). Somasundaran et al. (2007) develop genre-speci.c lexicons using interesting function word combinations for detecting opinions in meetings. Despite similarities, our work is different from subjectivity analysis because the later only discriminates between opinions and facts. A discussion sentence may display an opinion about some topic yet no attitude. The language constituents considered in opinion detection may be different from those used to detect attitude. Moreover, extracting attitudes from online discussions is different from targeting subjective expressions (Josef Ruppenhofer and Wiebe, 2008; Kim and Hovy, 2004). The later usually has a limited set of targets that compete for the subjective expressions (for example in movie review, targets could be: director, actors, plot, and so forth). We cannot use similar methods because we are working on an open domain where anything could be a target. A very detailed survey that covers techniques and approaches in sentiment analysis and opinion mining could be found in (Pang and Lee, 2008).

There is also some related work on mining online discussions. Lin et al (2009) proposes a sparse coding-based model simultaneously model semantics and structure of threaded discussions. Shen et al (2006) proposes three clustering methods for exploiting the temporal information in the streams, as well as an algorithm based on linguistic features to analyze the discourse structure information. Huang et al (2007) used an SVM classifier to extract (thread-title, reply) pairs as chat knowledge from online discussion forums to support the construction of a chatbot for a certain domain. Other work has focused on the structure of questions and question-answer pairs in online forums and discussions (Ding et al., 2008; Cong et al., 2008).

3 Problem Definition

Assume we have a set of sentences exchanged between participants in an online discussion. Our objective is to identify sentences that display an attitude from the text writer to the text recepient from those that do not. An attitude is the mental position of one particpant with regard to another participant. An attitude may not be directly observable, but rather inferred from what particpants say to one another. The attitude could be either positive or negative. Strategies for showing a positive attitude may include agreement, and praise, while strategies for showing a negative attitude may include disagreement, insults, and negative slang. After identifying sentences that display an attitude, we also predict the sign (positive or negative) of that attitude.

4 Approach

In this section, we describe a model which, given a sentence, predicts whether it carries an attitude from the text writer toward the text recipient or not. Any given piece of text exchanged between two participants in a discussion could carry an attitude toward the text recipient, an attitude towards the topic, or no attitude at all. As we are only interested in attitudes between participants, we limit our study to sentences that use second person pronouns. Second person pronouns are usually used in conversational genre to indicate that the text writer is addressing the text recipient. After identifying those sentences, we do some pre-processing to extract the most relevant fragments. We examine these fragments to to identify the polarity of every word in the sentence. Every word could be assigned a semantic orientation. The semantic orientation could be either positive, negative, or neutral. The existence of polarized words in any sentence is an important indicator of whether it carries an attitude or not.

The next step is to extract several patterns at different levels of generalization representing any given sentence. We use those patterns to build two Markov models for every kind of patterns. The first model characterizes the relation between different tokens for all patterns that correspond to sentences that have an attitude. The second model is similar to the first one, but it uses all patterns that correspond to sentences that do not have an attitude. Given a new sentence, we extract the corresponding patterns and estimate the likelihood of every pattern being generated from the two corresponding models. We then compare the likelihood of the sentence under the two models and use this as a feature to predict the existence of an attitude. A pair of models will be built for every kind of patterns. If we have n different patterns, we will have n different likelihood ratios that come from n pairs of models.

4.1 Word Polarity Identification

Identifying the polarity of words is an important step for our method. Our word identification module is similar to the work in (Annon, 2010). We construct a graph where each node represent a word/part-of-speech pair. Two nodes are linked if the words are related. We use WordNet (Miller, 1995) to link related words based on synonyms, hypernyms, and similar to relations. For words that do not appear in Wordnet, we used Wiktionary, a collaboratively constructed dictionary. We also add some links based on co-occurrence statistics between words as from a large corpus. The resulting graph is a graph $G(W, E)$ where W is a set of word/part-of-speech pairs, and E is the set of edges connecting related words.

We define a random walk model on the graph, where the set of nodes correspond to the state space of the random walk. Transition probabilities are calculated by normalizing the weights of the edges out of every node. Let $S+$ and $S-$ be two sets of vertices representing seed words that are already labeled as either positive or negative respectively. We used the list of labeled seeds from (Hatzivassiloglou and McKeown, 1997) and (Stone et al., 1966). For any given word w, we calculate the mean hitting time between w, and the two seed sets $h(w|S+)$, and $h(w|S-)$. The mean hitting time $h(i|k)$ is defined as the average number of steps a random walker, starting in state $i \neq k$, will take to enter state k for the first time (Norris, 1997). If $h(w|S+)$ is greater than $h(w|S-)$, the word is classified as negative, otherwise it is classified as positive. We also use the method described in (Wilson et al., 2005) to determine the contextual polarity of the identified words. The set of features used to predict contextual polarity include word, sentence, polarity, structure, and other features.

4.2 Identifying Relevant Parts of Sentences

The writing style in online discussion forums is very informal. Some of the sentence are very long, and punctuation marks are not always properly used. To solve this problem, we decided to use the grammatical structure of sentences to identify the most relevant part of sentences that would be the subject of further analysis. Figure 1 shows a parse tree representing the grammatical structure of a particular sentence. If we closely examine the sentence, we will notice that we are only interested in a part of the sentence that includes the second person pronoun ”you“. We extract this part, by starting at the word of interest , in this case ”you“, and go up in the hierarchy till we hit the first sentence clause. Once, we reach a sentence clause, we extract the corresponding text if it is grammatical, otherwise we go up one more level to the closest sentence clause. We used the Stanford parser to generate the grammatical structure of sentences (Klein and Manning, 2003).

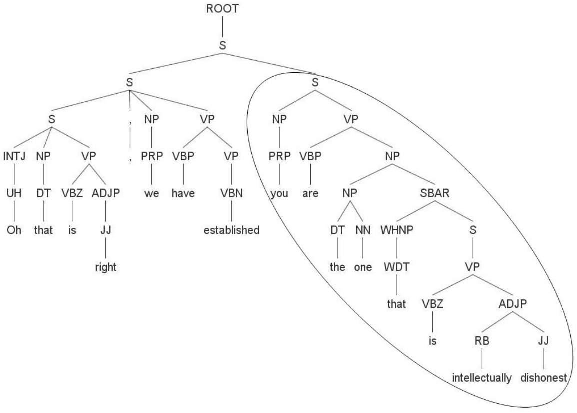


Figure 1: An example showing how to identify the relevant part of a sentence.

4.3 Sentences as Patterns

The fragments we extracted earlier are more relevant to our task and are more suitable for further analysis. However, these fragments are completely lexicalized and consequently the performance of any analysis based on them will be limited by data sparsity. We can alleviate this by using more general representations of words. Those general representations can be used a long with words to generate a set of patterns that represent each fragment. Each pattern consists of a sequence of tokens. Examples of such patterns could use lexical items, part-of-speech (POS) tags, word polarity tags, and dependency relations.

We use three different patterns to represent each fragments:

- Lexical patterns: All polarized words are replaces with the corresponding polarity tag, and all other words are left as is.
- Part-of-speech patterns: All words are replaced with their POS tags. Second person pronouns are left as is. Polarized words are replaced with their polarity tags and their POS tags.
- Dependency grammar patterns: the shortest path connecting every second person pronoun

to the closed polarized word is extracted. The second person pronoun, the polarized word tag, and the types of the dependency relations along the path connecting them are used as a pattern. It has been shown in previous work on relation extraction that the shortest path between any two entities captures the the information required to assert a relationship between them (Bunescu and Mooney, 2005). Every polarized word is assigned to the closest second person pronoun in the dependency tree. This is only useful for sentences that have polarized words.

Table 1 shows the different kinds of representations for a particular sentence. We use text, part-of-speech tags, polarity tags, and dependency relations. The corresponding patterns for this sentence are shown in Table 2.

4.4 Building the Models

Given a set of patterns representing a set of sentences, we can build a graph $G = V, E, w$ where V is the set of all possible token that may appear in the patterns. $E = V \times V$ is the set of possible transitions between any two tokens. $w : E \rightarrow [0..1]$ is a weighting function that assigns to every pair of states (i, j) a weight $w(i, j)$ representing the probability that we have a transition from state i to state j.

This graph corresponds to a Markovian model. The set of states are the vocabulary, and the the transition probabilities between states are estimated using Maximum Likelihood estimation as follows:

$$P_{ij} = \frac{N_{ij}}{N_i}$$

where N_{ij} is the number of times we saw a transition from i to state j, and N_i is the total number of times we saw state i in the training data. This is similar to building a language model over the language of the patterns.

We build two such models for every kind of patterns. The first model is built using all sentences that appeared in the training dataset and was labeled as having an attitude, and the second model is built using all sentences in the training dataset that do not have an attitude. If we have n kinds of patterns, we will build one such pair for every kind of patterns. Hence, we will end up with $2n$ models.

4.5 Identifying Sentences with Attitude

We split our training data into two splits; the first containing all sentences that have an attitude and the second containing all sentences that do not have an attitude. Given the methodology described in the previous section, we build n pairs of Markov models. Given any sentence, we extract the corresponding patterns and estimate the log likelihood that this sequence of tokens was generated from every model.

Given a model M, and sequence of tokens $T = (T_1, T_2, \ldots TS_n)$, the probability of this token sequence being generated from M is:

$$P_M(T) = \prod_{i=2}^{n} P(T_i|T_1, \ldots, T_{i-1}) = \prod_{i=2}^{n} W(T_{i-1}, T_i)$$

where n is the number of tokens in the pattern, and W is the probability transition function.

The log likelihood is then defined as:

$$LL_M(T) = \sum_{i=2}^{n} \log W(T_{i-1}, T_i)$$

For every pair of models, we may use the ratio between the two likelihoods as a feature:

$$f = \frac{LL_{M_{att}}(T)}{LL_{M_{noatt}}(T)}$$

where T is the token sequence, $LL_{M_{att}}(T)$ is the log likelihood of the sequence given the attitude model, and $LL_{M_{noatt}}(T)$ is the log likelihood of the pattern given the no-attitude model.

Given the n kinds of patterns, we can calculate three different features. A standard machine learning classifier is then trained using those features to predict whether a given sentence has an attitude or not.

4.6 Identifying the Sign of an Attitude

To determine the orientation of an attitude sentence, we tried two different methods. The first method assumes that the orientation of an attitude sentence is directly related to the polarity of the words it contains. If the sentence has only positive and neutral

Table 1: Tags used for building patterns

Text	That makes your claims so ignorant
POS	That/DT makes/VBZ your/PRP$ claims/NNS so/RB ignorant/JJ
Polarity	That/O makes/O your/O claims/O so/O ignorant/NEG
Dependency	your $\xrightarrow{\text{poss}}$ claims $\xrightarrow{\text{nsubj}}$ ignorant

Table 2: Sample patterns

Lexical pattern	That makes your claims so NEG
POS pattern	DT VBZ your_PRP$ NNS RB NEG_JJ
Dependency pattern	your poss nsubj NEG

words, it is classified as positive. If the sentence has only negative and neutral words, it is classified as negative. If the sentence has both positive and negative words, we calculate the summation of the polarity scores of all positive words and that of all negative words. The polarity score of a word is an indicator of how strong of a polarized word it is. If the former is greater, we classify the sentence as positive,otherwise we classify the sentence as negative.

The problem with this method is that it assumes that all polarized words in a sentence with an attitude target the text recipient. Unfortunately, that is not always correct. For example, the sentence ”You are completely unqualified to judge this great person” has a positive word ”great” and a negative word ”unqualified”. The first method will not be able to predict whether the sentence is positive or negative. To solve this problem, we use another method that is based on the paths that connect polarized words to second person pronouns in a dependency parse tree. For every positive word w , we identify the shortest path connecting it to every second person pronoun in the sentence then we compute the average length of the shortest path connecting every positive word to the closest second person pronoun. We repeat for negative words and compare the two values. The sentence is classified as positive if the average length of the shortest path connecting positive words to the closest second person pronoun is smaller than the corresponding value for negative words. Otherwise, we classify the sentence as negative.

5 Data

Our data was randomly collected from a set of discussion groups. We collected a large number of threads from the first quarter of 2009 from a set of Usenet discussion groups. All threads were in English, and had 5 posts or more. We parsed the downloaded threads to identify the posts and senders. We kept posts that have quoted text and discarded all other posts. The reason behind that is that participants usually quote other participants text when they reply to them. This restriction allows us to identify the target of every post, and raises the probability that the post will display an attitude from its writer to its target. We plan to use more sophsticated methods for reconstructing the reply structure like the one in (Lin et al., 2009). From those posts, we randomly selected approximately 10,000 sentences that use second person pronouns. We explained earlier how second person pronouns are used in discussions genres to indicate the writer is targeting the text recipient. Given a random sentence selected from some random discussion thread, the probability that the sentence does not have an attitude is significantly larger than the probability that it will have an attitude. Hence, restricting our dataset to posts with quoted text and sentences with second person pronouns is very important to make sure that we will have a considerable amount of attitudinal sentences. The data was tokenized, sentence-split, part-of-speech tagged with the OpenNLP toolkit. It was parsed with the Stanford dependency parser (Klein and Manning, 2003).

5.1 Annotation Scheme

The goals of the annotation scheme are to distinguish sentences that display an attitude from those that do not. Sentences could display either a negative or a positive attitude. Disagreement, insults, and negative slang are indicators of negative attitude.

	A	B	C	D
A	-	82.7	80.6	82.1
B	81.0	-	81.9	82.9
C	77.8	78.2	-	83.8
D	78.3	77.7	78.6	-

Table 3: Inter-annotator agreement

Agreement, and praise are indicators of positive attitude. Our annotators were instructed to read every sentence and assign two labels to it. The first specifies whether the sentence displays an attitude or not. The existence of an attitude was judged on a three point scale: attitude, unsure, and no-attitude. The second is the sign of the attitude. If an attitude exists, annotators were asked to specify whether the attitude is positive or negative. To evaluate inter-annotator agreement, we use the agr operator presented in (Wiebe et al., 2005). This metric measures the precision and recall of one annotator using the annotations of another annotator as a gold standard. The process is repeated for all pairs of annotators, and then the harmonic mean of all values is reported. Formally:

$$agr(A|B) = \frac{|A \cap B|}{|A|} \tag{1}$$

where A, and B are the annotation sets produced by the two reviewers. Table 3 shows the value of the agr operator for all pairs of annotators. The harmonic mean of the agr operator is 80%. The agr operator was used over the Kappa Statistic because the distribution of the data was fairly skewed.

6 Experiments

6.1 Experimental Setup

We performed experiments on the data described in the previous section. The number of sentences with an attitude was around 20% of the entire dataset. The class imbalance caused by the small number of attitude sentences may hurt the performance of the learning algorithm (Provost, 2000). A common way of addressing this problem is to artificially rebalance the training data. To do this we down-sample the majority class by randomly selecting, without replacement, a number of sentences without an attitude that equals the number of sentences with an attitude. That resulted in a balanced subset, approximately 4000 sentences, that we used in our experiments.

We used Support Vector Machines (SVM) as a classifier. We optimized SVM separately for every experiment. We used 10-fold cross validation for all tests. We evaluate our results in terms of precision, recall, accuracy, and F1. Statistical significance was tested using a 2-tailed paired t-test. All reported results are statistically significant at the 0.05 level. We compare the proposed method to several other baselines that will be described in the next subsection. We also perform experiments to measure the performance if we mix features from the baselines and the proposed method.

6.2 Baselines

The first baseline is based on the hypothesis that the existence of polarized words is a strong indicator that the sentence has an attitude. As a result, we use the number of polarized word in the sentence, the percentage of polarized words to all other words, and whether the sentences has polarized words with mixed or same sign as features to train an SVM classifier to detect attitude.

The second baseline is based on the proximity between the polarized words and the second person pronouns. We assume that every polarized word is associated with the closest second person pronoun. Let w be a polarized word and $p(w)$ be the closes second person pronoun, and $surf\_dist(w, p(w))$ be the surface distance between w and $p(w)$. This baseline uses the minimum, maximum, and average of $surf\_dist(w, p(w))$ for all polarized words as features to train an SVM classifier to identify sentences with attitude.

The next baseline uses the dependency tree distance instead of the surface distance. We assume that every polarized word is associated to the second person pronoun that is connected to it using the smallest shortest path. The $dep\_dist(w, p(w))$ is calculated similar to the previous baselines but using the dependency tree distance. The minimum, maximum, and average of this distance for all polarized words are used as features to train an SVM classifier.

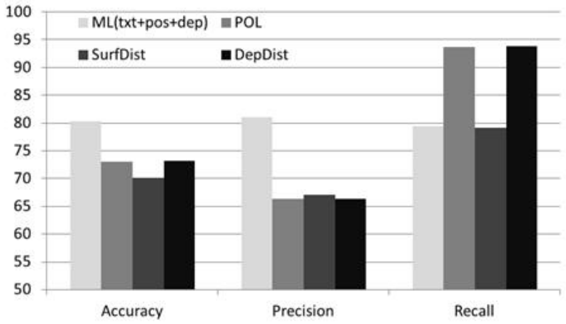


Figure 2: Accuracy, Precision, and Recall for the Proposed Approach and the Baselines.

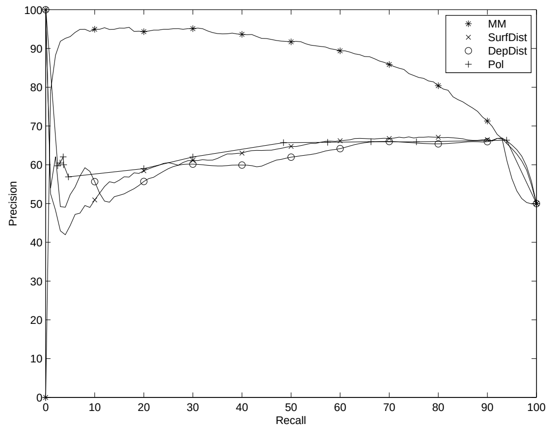


Figure 3: Precision Recall Graph.

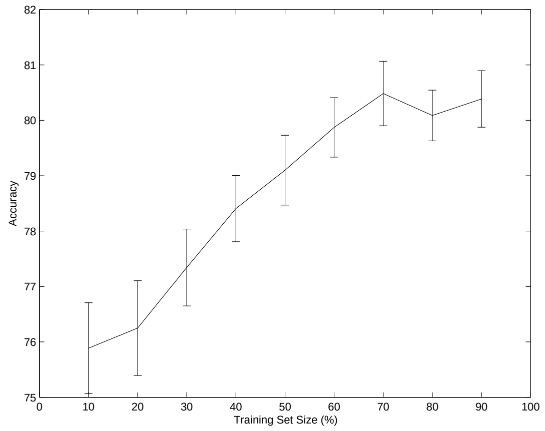


Figure 4: Accuracy Learning Curve for the Proposed Method.

6.3 Results and Discussion

Figure 2 compares the accuracy, precision, and recall of the proposed method (ML), the polarity based classifier (POL), the surface distance based classifier (Surf_Dist), and the dependency distance based classifier (Dep_Dist). The values are selected to optimize F1. The figure shows that the surface distance based classifier behaves poorly with low accuracy, precision, and recall. The two other baselines behave poorly as well in terms of precision and accuracy, but they do very well in terms of recall. We looked at some of the examples to understand why those two baselines achieve very high recall. It turns out that they tend to predict most sentences that have polarized words as sentences with attitude. This results in many false positives and low true negative rate. Achieving high recall at the expense of losing precision is trivial. On the other hand, we notice that the proposed method results in very close values of precision and recall at the optimum F1 point.

To better compare the performance of the proposed method and the baseline, we study the the precision-recall curves for all methods in Figure 3. We notice that the proposed method outperforms all baselines at all operating points. We also notice that the proposed method provides a nice trade-off between precision and recall. This allows us some flexibility in choosing the operating point. For example, in some applications we might be interested in very high precision even if we lose recall, while in other applications we might sacrifice precision in order to get high recall. On the other hand, we notice that the baselines always have low precision regardless of recall.

Table 4 shows the accuracy, precision, recall, and F1 for the proposed method and all baselines. It also shows the performance when we add features from the baselines to the proposed method, or merge some of the baselines. We see that we did not get any improvement when we added the baseline features to the proposed method. We believe that the proposed method captures all the information captured by the baselines and more.

Our proposed method uses three different features that correspond to the three types of patterns we use to represent every sentence. To understand the contributions of every feature, we measure the performance of every feature by itself and also all possible combinations of pairs of features. We compare that to the performance we get when using all features in Table 5. We see that the part-of-speech patterns performs better than the text patterns. This makes sense because the former suffers from data sparsity. Dependency patterns performs best in terms of recall, while part-of-speech patterns outperform all others in terms of precision, and accuracy. All pairs of features outperform any single feature that belong to the corresponding pair in terms of F1. We also notice that using the three features results in better performance when compared to all other combinations. This shows that every kind of pattern captures slightly different information when compared to the others. It also shows that merging the three features improves performance.

One important question is how much data is required to the proposed model. We constructed a learning curve, shown in Figure 4, by fixing the test set size at one tenth of the data, and varying the training set size. We carried out ten-fold cross validation as with our previous experiments. We see that adding more data continues to increase the accuracy, and that accuracy is quite sensitive to the training data. This suggests that adding more data to this model could lead to even better results.

We also measured the accuracy of the two methods we proposed for predicting the sign of attitudes. The accuracy of the first model that only uses the count and scores of polarized words was 95%. The accuracy of the second method that used dependency distance was 97%.

6.4 Error Analysis

We had a closer look at the results to find out what are the reasons behind incorrect predictions. We found two main reasons. First, errors in predicting word polarity usually propagates and results in errors in attitude prediction. The reasons behind incorrect word polarity predictions is ambiguity in word senses and infrequent words that have very few connection in thesaurus. A possible solution to this type of errors is to improve the word polarity identification module by including word sense disambiguation and adding more links to the words graph using glosses or co-occurrence statistics. The second reason is that some sentences are sarcastic in nature. It is so difficult to identify such sentences. Identifying sarcasm should be addressed as a separate problem. A method that utilizes holistic approaches that takes context and previous interactions between discussion participants into consideration could be used to address it.

Method	Accuracy	Precision	Recall	F1
ML	80.3	81.0	79.4	80.2
POL	73.1	66.4	93.9	77.7
ML+POL	79.9	77.9	83.4	80.5
SurfDist	70.2	67.1	79.2	72.7
DepDist	73.1	66.4	93.8	77.8
SurfDist+ DepDist	73.1	66.4	93.8	77.7
ML+SurfDist	73.9	67.2	93.6	78.2
ML+DepDist	72.8	66.1	93.8	77.6
ML+SurfDist+ DepDist	74.0	67.2	93.4	78.2
SurfDist+ DepDist+POL	73.1	66.3	93.8	77.7
ML+SurfDist+ DepDist+POL	73.0	66.2	93.8	77.6

Table 4: Precision, Recall, F1, and Accuracy for the proposed method, the baselines, and different combinations of proposed method and the baselines features

Method	Accuracy	Precision	Recall	F1
txt	75.5	74.1	78.6	76.2
pos	77.7	78.2	76.9	77.5
dep	74.7	70.4	85.1	77.0
txt+pos	77.8	77.0	79.4	78.1
txt+dep	79.4	79.6	79.2	79.4
pos+dep	80.4	79.1	82.5	80.7
txt+pos+dep	80.3	81.0	79.4	80.2

Table 5: Precision, Recall, F1, and Accuracy for different combinations of the proposed method's features.

7 Conclusions

We have shown that training a supervised Markov model of text, part-of-speech, and dependecy patterns allows us to identify sentences with attitudes from sentences without attitude. This model is more accurate than several other baselines that use features based on the existence of polarized word, and proximity between polarized words and second person pronouns both in text and dependecy trees. This method allows to extract signed social networks from multi-party online discussions. This opens the door to research efforts that go beyond standard social network analysis that is based on positve links only. It also allows us to study dynamics behind interactions in online discussions, the relation between text and social interactions, and how groups form and break in online discussions.

Acknowledgments

This research was funded by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), through the U.S. Army Research Lab. All statements of fact, opinion or conclusions contained herein are those of the authors and should not be construed as representing the official views or policies of IARPA, the ODNI or the U.S. Government.

References

Alina Andreevskaia and Sabine Bergler. 2006. Mining wordnet for fuzzy sentiment: Sentiment tag extraction from wordnet glosses. In *EACL'06*.

Carmen Banea, Rada Mihalcea, and Janyce Wiebe. 2008. A bootstrapping method for building subjectivity lexicons for languages with scarce resources. In *LREC'08*.

Razvan C. Bunescu and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *HLT '05*, pages 724–731, Morristown, NJ, USA. Association for Computational Linguistics.

Gao Cong, Long Wang, Chin-Yew Lin, Young-In Song, and Yueheng Sun. 2008. Finding question-answer pairs from online forums. In *SIGIR '08*, pages 467–474.

Shilin Ding, Gao Cong, Chin-Yew Lin, and Xiaoyan Zhu. 2008. Using conditional random fields to extract contexts and answers of questions from online forums. In *ACL'08*, pages 710–718.

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *EACL'97*, pages 174–181.

Vasileios Hatzivassiloglou and Janyce Wiebe. 2000. Effects of adjective orientation and gradability on sentence subjectivity. In *COLING*, pages 299–305.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *KDD'04*, pages 168–177.

Jizhou Huang, Ming Zhou, and Dan Yang. 2007. Extracting chatbot knowledge from online discussion forums. In *IJCAI'07*, pages 423–428.

Swapna Somasundaran Josef Ruppenhofer and Janyce Wiebe. 2008. Finding the sources and targets of subjective expressions. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*.

Jaap Kamps, Maarten Marx, Robert J. Mokken, and Maarten De Rijke. 2004. Using wordnet to measure semantic orientations of adjectives. In *National Institute for*, pages 1115–1118.

Hiroshi Kanayama and Tetsuya Nasukawa. 2006. Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *EMNLP'06*, pages 355–363.

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *COLING*, pages 1367–1373.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *ACL'03*, pages 423–430.

Jérôme Kunegis, Andreas Lommatzsch, and Christian Bauckhage. 2009. The slashdot zoo: mining a social network with negative edges. In *WWW'09*, pages 741–750, New York, NY, USA.

Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. 2010. Predicting positive and negative links in online social networks. In *WWW '10*, pages 641–650, New York, NY, USA. ACM.

Chen Lin, Jiang-Ming Yang, Rui Cai, Xin-Jing Wang, and Wei Wang. 2009. Simultaneously modeling semantics and structure of threaded discussions: a sparse coding approach and its applications. In *SIGIR '09*, pages 131–138.

George A. Miller. 1995. Wordnet: a lexical database for english. *Commun. ACM*, 38(11):39–41.

Satoshi Morinaga, Kenji Yamanishi, Kenji Tateishi, and Toshikazu Fukushima. 2002. Mining product reputations on the web. In *KDD'02*, pages 341–349.

Tetsuya Nasukawa and Jeonghee Yi. 2003. Sentiment analysis: capturing favorability using natural language processing. In *K-CAP '03: Proceedings of the 2nd international conference on Knowledge capture*, pages 70–77.

J. Norris. 1997. Markov chains. Cambridge University Press.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Ana-Maria Popescu and Oren Etzioni. Extracting product features and opinions from reviews. In *HLT-EMNLP'05*.

Foster Provost. 2000. Machine learning from imbalanced data sets 101. In *Proceedings of the AAAI Workshop on Imbalanced Data Sets*.

Ellen Riloff and Janyce Wiebe. 2003. Learning extraction patterns for subjective expressions. In *EMNLP'03*, pages 105–112.

Dou Shen, Qiang Yang, Jian-Tao Sun, and Zheng Chen. 2006. Thread detection in dynamic text message streams. In *SIGIR '06*, pages 35–42.

Swapna Somasundaran, Josef Ruppenhofer, and Janyce Wiebe. 2007. Detecting arguing and sentiment in meetings. In *Proceedings of the SIGdial Workshop on Discourse and Dialogue*.

Philip Stone, Dexter Dunphy, Marchall Smith, and Daniel Ogilvie. 1966. The general inquirer: A computer approach to content analysis. *The MIT Press*.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Extracting semantic orientations of words using spin model. In *ACL'05*, pages 133–140.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from Congressional floor-debate transcripts. In *EMNLP 2006*, pages 327–335.

Peter Turney and Michael Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems*, 21:315–346.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 1(2):0.

Janyce Wiebe. 2000. Learning subjective adjectives from corpora. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 735–740.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT/EMNLP'05*, Vancouver, Canada.

Hong Yu and Vasileios Hatzivassiloglou. 2003. Towards answering opinion questions: separating facts from opinions and identifying the polarity of opinion sentences. In *EMNLP'03*, pages 129–136.

Hashing-based Approaches to Spelling Correction of Personal Names

Raghavendra Udupa
Microsoft Research India
Bangalore, India
raghavu@microsoft.com

Shaishav Kumar
Microsoft Research India
Bangalore, India
v-shaisk@microsoft.com

Abstract

We propose two hashing-based solutions to the problem of fast and effective personal names spelling correction in People Search applications. The key idea behind our methods is to learn hash functions that map similar names to similar (and compact) binary codewords. The two methods differ in the data they use for learning the hash functions - the first method uses a set of names in a given language/script whereas the second uses a set of bilingual names. We show that both methods give excellent retrieval performance in comparison to several baselines on two lists of misspelled personal names. More over, the method that uses bilingual data for learning hash functions gives the best performance.

1 Introduction

Over the last few years, People Search has emerged as an important search service. Unlike general Web Search and Enterprise Search where users search for information on a wide range of topics including people, products, news, events, etc., People Search is about people. Hence, personal names are used predominantly as queries in People Search. As in general Web Search, a good percentage of queries in People Search is misspelled. Naturally, spelling correction of misspelled personal names plays a very important role in not only reducing the time and effort needed by users to find people they are searching for but also in ensuring good user experience.

Spelling errors in personal names are of a different nature compared to those in general text. Long before People Search became widely popular, researchers working on the problem of personal name matching had recognized the human tendency to be inexact in recollecting names from the memory and specifying them. A study of personal names in hospital databases found that only 39% of the errors in the names were single typographical errors (Friedman and Sideli, 1992)[1]. Further, multiple and long distance typographical errors (*Gregzorz Kondrak* for *Grzegorz Kondrak*), phonetic errors (as in *Erik Bryl* for *Eric Brill*), cognitive errors (as in *Silvia Cucerzan* for *Silviu Cucerzan*) and word substitutions (as in *Rob Moore* for *Bob Moore*) are observed relatively more frequently in personal names compared to general text.

In addition to within-the-word errors, People Search queries are plagued by errors that are not usually seen in general text. The study by Friedman and Sideli discovered that 36% of the errors were due to addition or deletion of a word (as in *Ricardo Baeza* for *Ricardo Baeza-Yates*) (Friedman and Sideli, 1992). Although word addition and deletion generally do not come under the purview of spelling correction, in People Search they are important and need to be addressed.

Standard approaches to general purpose spelling correction are not well-suited for correcting misspelled personal names. As pointed out by (Cucerzan and Brill, 2004), these approaches either try to correct individual words (and will fail to correct *Him Clijsters* to *Kim Clijsters*) or employ features based on relatively wide context windows

[1] In contrast, 80% of misspelled words in general text are due to single typographical errors as found by (Damerau, 1964).

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1256–1265,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

which are not available for queries in Web Search and People Search. Spelling correction techniques meant for general purpose web-queries require large volumes of training data in the form of query logs for learning the error models (Cucerzan and Brill, 2004), (Ahmad and Kondrak, 2005). However, query logs are not available in some applications (e.g. Email address book search). Further, unlike general purpose web-queries where word order often matters, in People Search word order is lax (e.g. I might search for either *Kristina Toutanova* or *Toutanova Kristina*). Therefore, spelling correction techniques that rely crucially on bigram and higher order language models will fail on queries with a different word order than what is observed in the query log.

Unlike general purpose Web Search where it is not reasonable to assume the availability of a high-coverage trusted lexicon, People Search typically employs large authoritative name directories. For instance, if one is searching for a friend on Facebook, the correct spelling of the friend's name exists in the Facebook people directory[2] (assuming that the friend is a registered user of Facebook at the time of the search). Similarly, if one is searching for a contact in Enterprise address book, the correct spelling of the contact is part of the address book. In fact, even in Web Search, broad-coverage name directories are available in the form of Wikipedia, IMDB, etc. The availability of large authoritative name directories that serve as the source of trusted spellings of names throws open the possibility of correcting misspelled personal names with the help of name matching techniques (Pfeifer et al., 1996), (Christen, 2006), (Navarro et al., 2003). However, the best of the name matching techniques can at best work with a few thousand names to give acceptable response time and accuracy. They do not scale up to the needs of People Search applications where the directories can have millions of names.

In this work, we develop hashing-based name similarity search techniques and employ them for spelling correction of personal names. The motivation for using hashing as a building block of spelling correction is the following: given a query, we want to return the global best match in the name directory that exceeds a similarity threshold. As matching the query with the names in the directory is a time consuming task especially for large name directories, we solve the search problem in two stages:

- NAME_BUCKETING: For each token of the query, we do an approximate nearest neighbor search of the name tokens of the directory and produce a list of candidates, i.e., tokens that are approximate matches of the query token. Using the list of candidate tokens, we extract the list of candidate names which contain at least one approximately matching token.

- NAME_MATCHING: We do a rigorous matching of the query with candidate names.

Clearly, our success in finding the right name suggestion for the query in the NAME_MATCHING stage depends crucially on our success in getting the right name suggestion in the list of candidates produced by the NAME_BUCKETING stage search. Therefore, we need a name similarity search technique that can ensure very high recall without producing too many candidates. Hashing is best suited for this task of fast and approximate name matching. We hash the query tokens as well as directory tokens into d bit binary codes. With binary codes, finding approximate matches for a query token is as easy as finding all the database tokens that are at a Hamming distance of r or less from the query token in the binary code representation (Shakhnarovich et al., 2008), (Weiss et al., 2008). When the binary codes are compact, this search can be done in a fraction of a second on directories containing millions of names on a simple processor.

Our contributions are:

- We develop a novel data-driven technique for learning hash functions for mapping similar names to similar binary codes using a set of names in a given language/script (i.e. monolingual data). We formulate the problem of learning hash functions as an optmization problem whose relaxation can be solved as a generalized Eigenvalue problem. (Section 2.1).

- We show that hash functions can also be learnt using bilingual data in the form of name equivalents in two languages. We formulate the

[2] http://www.facebook.com/directory/people/

problem of learning hash functions as an optmization problem whose relaxation can be solved using Canonical Correlation Analysis. (Section 2.2)

- We develop new similarity measures for matching names (Section 3.1).
- We evaluate the two methods systematically and compare our performance against multiple baselines. (Section 5).

2 Learning Hash Functions

In this section, we develop two techniques for learning hash functions using names as training data. In the first approach, we use monolingual data consisting of names in a language whereas in the second we use bilingual name pairs. In both techniques, the key idea is the same: we learn hash functions that map similar names in the training data to similar codewords.

2.1 M-HASH: Learning with Monolingual Names Data

Let (s, s') be a pair of names and $w(s, s')$ be their similarity[3]. We are given a set of name pairs $T = \{(s, s')\}$ as the training data. Let $\phi(s) \in R^{d_1}$ be the feature representation of s. We want to learn a hash function f that maps each name to a d bit codeword: $f : s \mapsto \{-1, 1\}^d$. We also want the Hamming distance of the codeword of s to the codeword of s' be small when $w(s, s')$ is large. Further, we want each bit of the codewords to be either 1 or -1 with equal probablity and the successive bits of the codewords to be uncorrelated. Thus we arrive at the following optimization problem[4]:

$$\begin{aligned} minimize: &\quad \sum_{(s,s')\in T} w\left(s, s'\right) \left\|f(s) - f\left(s'\right)\right\|^2 \\ s.t.: & \\ &\quad \sum_{s:(s,s')\in T} f(s) = 0 \\ &\quad \sum_{s:(s,s')\in T} f(s)\, f(s)^T = \rho^2 I_d \\ &\quad f(s), f\left(s'\right) \in \{-1, 1\}^d \end{aligned}$$

where I_d is an identity matrix of size $d \times d$.

Note that the second constraint helps us avoid the trap of mapping all names to the same codeword and thereby making the Hamming error zero while satisfying the first and last constraints.

It can be shown that the above minimization problem is NP-hard even for 1-bit codewords (Weiss et al., 2008). Further, the optimal solution gives codewords only for the names in the training data. As we want f to be defined for all s, we address the out-of-sample extension problem by relaxing f as follows[5]:

$$f_R(s) = A^T \phi(s) = \left(a_1^T \phi(s), \ldots, a_d^T \phi(s)\right)^T \quad (1)$$

where $A = [a_1, \ldots, a_d] \in R^{d_1 \times d}$ is a rank d matrix ($d \leq d_1$).

After the linear relaxation (Equation 1), the first constraint simply means that the data be centered, i.e., have zero mean. We center Φ by subtracting the mean of Φ from every $\phi(s) \in \Phi$ to get $\hat{\Phi}$.

Subsequent to the above relaxation, we get the following optimization problem:

$$\begin{aligned} minimize: &\quad Tr\ A^T \hat{\Phi} L \hat{\Phi}^T A && (2) \\ s.t.: & && (3) \\ &\quad A^T \hat{\Phi} \hat{\Phi}^T A = \rho^2 I_d && (4) \end{aligned}$$

where L is the graph Laplacian for the similarity matrix W defined by the pairwise similarities $w(s, s')$.

The minimization problem can be transformed into a generalized Eigenvalue problem and solved efficiently using either Cholesky factorization or QZ algorithm (Golub and Van Loan, 1996):

$$\hat{\Phi} L \hat{\Phi}^T A = \hat{\Phi} \hat{\Phi}^T A \Lambda \quad (5)$$

where Λ is a $d \times d$ diagonal matrix.

Once A has been estimated from the training data, the codeword of a name s can be produced by binarizing each coordinate of $f_R(s)$:

$$f(s) = \left(sgn\left(a_1^T \phi(s)\right), \ldots, sgn\left(a_d^T \phi(s)\right)\right)^T \quad (6)$$

where $sgn(u) = 1$ if $u > 0$ and -1 otherwise for all $u \in R$.

[3] We used 1− length normalized Edit Distance between s and s' as $w(s, s')$.

[4] Note that the Hamming distance of a codeword y to another codeword y' is $\frac{1}{4}\left\|y - y'\right\|^2$.

[5] In contrast to our approach, Spectral Hashing, a well-known hashing technique, makes the unrealistic assumption that the training data is sampled from a multidimensional uniform distribution to address the out-of-sample extension problem (Weiss et al., 2008).

In the reminder of this work, we call the system that uses the hash function learnt from monolingual data as *M-HASH*.

2.2 B-HASH: Learning with Bilingual Names Data

Let (s, t) be a pair of name s and its transliteration equivalent t in a different language/script. We are given the set $T = \{(s, t)\}$ as the training data. Let $\phi(s) \in R^{d_1}$ (and resp. $\psi(t) \in R^{d_2}$) be the feature representation of s (and resp. t). We want to learn a pair of hash functions f, g that map names to d bit codewords: $f : s \mapsto \{-1, 1\}^d$, $g : t \mapsto \{-1, 1\}^d$. We also want the Hamming distance of the codeword of a name to the codeword of its transliteration be small. As in Section 2.1, we want each bit of the codewords to be either 1 or -1 with equal probablity and the successive bits of the codewords to be uncorrelated. Thus we arrive at the following optimization problem:

$$\begin{aligned}
minimize : \quad & \sum_{(s,t) \in T} \| f(s) - g(t) \|^2 \\
s.t. : \quad & \\
& \sum_{s:(s,t) \in T} f(s) = 0 \\
& \sum_{t:(s,t) \in T} g(t) = 0 \\
& \sum_{s:(s,t) \in T} f(s) f(s)^T = \rho^2 I_d \\
& \sum_{t:(s,t) \in S} g(t) g(t)^T = \rho^2 I_d \\
& f(s), g(t) \in \{-1, 1\}^d
\end{aligned}$$

where I_d is an identity matrix of size $d \times d$.

As we want f (and resp. g) to be defined for all s (and resp. t), we relax f (and resp. g) as follows:

$$f_R(s) = A^T \phi(s) \tag{7}$$

$$g_R(t) = B^T \psi(s) \tag{8}$$

where $A = [a_1, \ldots, a_d] \in R^{d_1 \times d}$ and $B = [b_1, \ldots, b_d] \in R^{d_2 \times d}$ are rank d matrices.

As before, we center Φ and Ψ to get $\hat{\Phi}$ and $\hat{\Psi}$ respectively. Thus, we get the following optimization problem:

$$minimize : \quad Tr\ H\left(A, B; \hat{\Phi}, \hat{\Psi}\right) \tag{9}$$

$$s.t. : \tag{10}$$

$$A^T \hat{\Phi} \hat{\Phi}^T A = \rho^2 I_d \tag{11}$$

$$B^T \hat{\Psi} \hat{\Psi}^T B = \rho^2 I_d \tag{12}$$

where $H\left(A, B; \hat{\Phi}, \hat{\Psi}\right) = \left(A^T \hat{\Phi} - B^T \hat{\Psi}\right)\left(A^T \hat{\Phi} - B^T \hat{\Psi}\right)^T$.

The minimization problem can be solved as a generalized Eigenvalue problem:

$$\hat{\Phi} \hat{\Psi}^T B = \hat{\Phi} \hat{\Phi}^T A \Lambda \tag{13}$$

$$\hat{\Psi} \hat{\Phi}^T A = \hat{\Psi} \hat{\Psi}^T B \Lambda \tag{14}$$

where Λ is a $d \times d$ diagonal matrix. Further, Equations 13 and 14 find the canonical coefficients of $\hat{\Phi}$ and $\hat{\Psi}$ (Hardoon et al., 2004).

As with monolingual learning, we get the codeword of s by binarizing the coordinates of $f_R(s)$[6]:

$$f(s) = \left(sgn\left(a_1^T \phi(s)\right), \ldots, sgn\left(a_d^T \phi(s)\right)\right)^T \tag{15}$$

In the reminder of this work, we call the system that uses the hash function learnt from bilingual data as *B-HASH*.

3 Similarity Score

In this section, we develop new techniques for computing the similarity of names at token level as well as a whole. We will use these techniques in the NAME_MATCHING stage of our algorithm (Section 4.2.1).

3.1 Token-level Similarity

We use a logistic function over multiple distance measures to compute the similarity between name tokens s and s':

$$K(s, s') = \frac{1}{1 + e^{-\sum_i \alpha_i d_i(s, s')}}. \tag{16}$$

While a variety of distance measures can be employed in Equation 16, two obvious choices

[6] As a biproduct of bilingual learning, we can hash names in the second language using g:

$$g(t) = \left(sgn\left(b_1^T \psi(t)\right), \ldots, sgn\left(b_d^T \psi(t)\right)\right)^T$$

are the normalized Damerau-Levenshtein edit distance between s and s' and the Hamming distance between the codewords of s and s' $(= \|f(s) - f(s')\|)$. In our experiments, we found that the continuous relaxation $\|f_R(s) - f_R(s')\|$ was better than $\|f(s) - f(s')\|$ and hence we used it with Damerau-Levenshtein edit distance. We estimated α_1 and α_2 using a small held out set.

3.2 Multi-token Name Similarity

Let $Q = s_1 s_2 \ldots s_I$ and $D = s'_1 s'_2 \ldots s'_J$ be two multi-token names. To compute the similarity between Q and D, we first form a weighted bipartite graph with a node for each s_i and a node for each s'_j and set edge weight to $K\left(s_i, s'_j\right)$. We then compute the weight (κ_{max}) of the maximum weighted matching[7] in this graph. The similarity between Q and D is then computed as

$$K(Q, D) = \frac{\kappa_{max}}{|I - J + 1|}. \tag{17}$$

4 Spelling Correction using Hashing

In this section, we describe our algorithm for spelling correction using hashing as a building block.

4.1 Indexing the Name Directory

Given a name directory, we break each name into its constituent tokens and form a set of distinct name tokens. Using the name tokens and the original names, we build an inverted index which, for each name token, lists all the names that have the token as a constituent. Further, we hash each name token into a d bit codeword as described in Equation 6 (and resp. Equation 15) when using the hash function learnt on monolingual data (and resp. bilingual data) and store in a hash table.

4.2 Querying the Name Directory

Querying is done in two stages: NAME_BUCKETING and NAME_MATCHING.

4.2.1 Name Bucketing

Given a query $Q = s_1 s_2 \ldots s_I$, we hash each s_i into a codeword y_i and retrieve all codewords in the hash table that are at a Hamming distance of r or less from y_i. We rank the name tokens thus retrieved using the token level similarity score of Section 3.1 and retain only the top 100. Using the top tokens, we get all names which contain any of the name tokens as a constituent to form the pool of candidates $\mathcal{C}$ for the NAME_MATCHING stage.

4.2.2 Name Matching

First we find the best match for a query Q in the set of candidates $\mathcal{C}$ as follows:

$$D^* = \underset{D \in \mathcal{C}}{\operatorname{argmax}}\, K(Q, D). \tag{18}$$

Next we suggest D^* as the correction for Q if $K(Q, D^*)$ exceeds a certain empirically determined threshold.

5 Experiments and Results

We now discuss the experiments we conducted to study the retrieval performance of the two hashing-based approaches developed in the previous sections. Apart from evaluating the systems on test sets using different name directories, we were interested in comparing our systems with several baselines, understanding the effect of some of the choices we made (e.g. training data size, conjugate language) and comparative analysis of retrieval performance on queries of different complexity.

5.1 Experimental Setup

We tested the proposed hashing-based spelling correction algorithms on two test sets:

- DUMBTIONARY: 1231 misspellings of various names from *Dumbtionary*[8] and a name directory consisting of about $550,000$ names gleaned from the English Wikipedia. Each of the misspellings had a correct spelling in the name directory.

- INTRANET: 200 misspellings of employees taken from the search logs of the intranet of a large organization and a name directory

[7] In practice, a maximal matching computed using a greedy approach suffices since many of the edges in the bipartite graph have low weight.

[8] http://www.dumbtionary.com

consisting of about $150,000$ employee names. Each of the misspellings had a correct spelling in the name directory.

Table 1 shows the average edit distance of a misspelling from the correct name. Compared to DUMBTIONARY, the misspellings in INTRANET are more severe as the relatively high edit distance indicates. Thus, INTRANET represents very hard cases for spelling correction.

Test Set	**Average**	**Std. Dev.**
DUMBTIONARY	1.39	0.76
INTRANET	2.33	1.60

Table 1: Edit distance of a misspelling from the correct name.

5.1.1 Training

For M-HASH, we used 30,000 single token names in English (sampled from the list of names in the Internet Movie Database[9]) as training data and for B-HASH we used 14,941 parallel single token names in English-Hindi [10]. Each name was represented as a feature vector over character bigrams. Thus, the name token *Klein* has the bigrams $\{\cdot k, kl, le, ei, in, n\cdot\}$ as the features.

We learnt the hash functions from the training data by solving the generalized Eigenvalue problems of Sections 2.1 and 2.2. For both M-HASH and B-HASH we used the top 32 Eigenvectors to form the hash function resulting in a 32 bit representation for every name token[11].

5.1.2 Performance Metric

We measured the performance of all the systems using Precision@1, the fraction of names for which a correct spelling was suggested at Rank 1.

5.1.3 Baselines

The baselines are two popular search engines (S1 and S2), Double Metaphone (DM), a widely used phonetic search algorithm (Philips, 2000) and BM25, a very popular Information Retrieval algorithm (Manning et al., 2008). To use BM25 algorithm for spelling correction, we represented each name as a bag of bigrams and set the parameters K and b to 2 and 0.75 respectively.

5.2 Results

5.2.1 DUMBTIONARY

Table 2 compares the results of the hashing-based systems with the baselines on DUMBTIONARY. As the misspellings in DUMBTIONARY are relatively easier to correct, all the systems give reasonably good retrieval results. Nevertheless, the results of M-HASH and B-HASH are substantially better than the baselines. M-HASH reduced the error over the best baseline (S1) by 13.04% whereas B-HASH reduced by 46.17% (Table 6).

M-HASH	**B-HASH**	**S1**	**S2**	**DM**	**BM25**
87.93	**92.53**	86.12	79.33	78.95	84.70

Table 2: Precision@1 of the various systems on DUMBTIONARY.

To get a deeper understanding of the retrieval performance of the various systems, we studied queries of varying complexity of misspelling. Table 3 compares the results of our systems with S1 for queries that are at various edit distances from the correct names. We observe that M-HASH and B-HASH are better than S1 in dealing with relatively less severe misspellings. More interestingly, B-HASH is consistently and significantly better than S1 even when the misspellings are severe.

Distance	**M-HASH**	**B-HASH**	**S1**
1	96.18	**96.55**	89.59
2	81.79	**87.42**	75.76
3	44.07	**67.80**	59.65
4	21.05	**31.58**	29.42
5	0.00	**37.50**	0.00

Table 3: Precision@1 for queries at various edit distances on DUMBTIONARY.

5.2.2 INTRANET

For INTRANET, search engines could not be used as baselines and therefore we compare our systems

[9] http://www.imdb.com

[10] We obtained the names from the organizers of NEWS2009 workshop (http://www.acl-ijcnlp-2009.org/workshops/NEWS2009/pages/sharedtask.html).

[11] We experimented with codewords of various lengths and found that the 32 bit representation gave the best tradeoff between retrieval accuracy and speed.

with Double Metaphone and BM25 in Table 4. We observe that both M-HASH and B-HASH give signficantly better retrieval results than the baselines. M-HASH reduced the error by 36.20% over Double Metaphone whereas B-HASH reduced it by 51.73%. Relative to BM25, M-HASH reduced the error by 31.87% whereas B-HASH reduced it by 48.44%.

M-HASH	B-HASH	DM	BM25
70.65	**77.79**	54.00	56.92

Table 4: Precision@1 of the various systems on INTRANET.

Table 5 shows the results of our systems for queries that are at various edit distances from the correct names. We observe that the retrieval results for each category of queries are consistent with the results on DUMBTIONARY. As before, B-HASH gives signficantly better results than M-HASH.

Distance	M-HASH	B-HASH
1	82.76	**87.93**
2	57.14	**72.86**
3	34.29	**65.71**
4	38.46	**53.85**
5	6.67	**26.67**

Table 5: Precision@1 for queries at various edit distances on INTRANET.

Test Set	M-HASH	B-HASH
DUMBTIONARY	13.04	**46.17**
INTRANET	36.20	**51.73**

Table 6: Percentage error reduction over the best baseline.

5.2.3 Effect of Training Data Size

As both M-HASH and B-HASH are data driven systems, the effect of training data size on retrieval performance is important to study. Table 7 compares the results for systems trained with various amounts of training data on DUMBTIONARY. B-HASH trained with just 1000 name pairs gives 95.5% of the performance of B-HASH trained with 15000 name pairs. Similarly, M-HASH trained with 1000 names gives 98.5% of the performance of M-HASH trained with 30000 name pairs. This is probably because the spelling mistakes in DUMBTIONARY are relatively easy to correct.

Table 8 shows the results on INTRANET. We see that increase in the size of training data brings substantial returns for B-HASH. In contrast, M-HASH gives the best results at 5000 and does not seem to benefit from additional training data.

Size	M-HASH	B-HASH
1000	86.60	**88.34**
5000	87.36	**91.13**
10000	86.96	**92.53**
15000	87.19	**92.20**
30000	87.93	-

Table 7: Precision@1 on DUMBTIONARY as a function of training data size.

Size	M-HASH	B-HASH
1000	**66.04**	66.03
5000	70.65	**72.67**
10000	68.09	**75.26**
15000	68.60	**77.79**
30000	65.40	-

Table 8: Precision@1 on INTRANET as a function of training data size.

5.2.4 Effect of Conjugate Language

In Sections 5.2.1 and 5.2.2, we saw that bilingual data gives substantially better results than monolingual data. In the experiments with bilingual data, we used English-Hindi data for training B-HASH. A natural question to ask is what happens when we use someother language, say Hebrew or Russian or Tamil, instead of Hindi. In other words, does the retrieval performance, on an average, vary substantially with the conjugate language?

Table 9 compares the results on DUMBTIONARY when B-HASH was trained using English-Hindi, English-Hebrew, English-Russian, and English-Tamil data. We see that the retrieval results are good despite the differences in the script and language. Clearly, the source language (English in our experiments) benefits from being paired with any target language. However, some languages seem to give substantially better results than others when used as the conjugate language. For instance, Hindi as a conjugate for English seems to be better than Tamil. At the time of writing this paper, we do not know the reason for this behavior. We believe that a combination of factors including feature representation, training data, and language-specific confusion matrix need to be studied in greater depth to say anything conclusively about conjugate languages.

Conjugate	DUMBTIONARY	INTRANET
Hindi	92.53	77.79
Hebrew	91.30	71.68
Russian	89.42	64.94
Tamil	90.48	69.12

Table 9: Precision@1 of B-HASH for various conjugate languages.

5.2.5 Error Analysis

We looked at cases where either M-HASH or B-HASH (or both) failed to suggest the correct spelling. It turns out that in the DUMBTIONARY test set, for 81 misspelled names, both M-HASH and B-HASH failed to suggest the correct name at rank 1. Similarly, in the case of INTRANET test set, both M-HASH and B-HASH failed to suggest the correct name at rank 1 for 47 queries. This suggests that queries that are difficult for one system are also in general difficult for the other system. However, B-HASH was able to suggest correct names for some of the queries where M-HASH failed. In fact, in the INTRANET test set, whenever B-HASH failed, M-HASH also failed. And interestingly, in the DUMBTIONARY test set, the average edit distance of the query and the correct name for the cases where M-HASH failed to get the correct name in top 10 while B-HASH got it at rank 1 was 2.96. This could be because M-HASH attempts to map names with smaller edit distances to similar codewords.

Table 10 shows some interesting cases we found during error analysis. For the first query, M-HASH suggested the correct name whereas B-HASH did not. For the second query, both M-HASH and B-HASH suggested the correct name. And for the third query, B-HASH suggested the correct name whereas M-HASH did not.

Query	M-HASH	B-HASH
John Tiler	**John Tyler**	John Tilley
Ddear Dragba	**Didear Drogba**	**Didear Drogba**
James Pol	James Poe	**James Polk**

Table 10: Error Analysis.

5.3 Query Response Time

The average query response time is a measure of the speed of a system and is an important factor in real deployments of a Spelling Correction system. Ideally, one would like the average query response time to be as small as possible. However, in practice, average query response time is not only a function of the algorithm's computational complexity but also the computational infrastructure supporting the system. In our expriments, we used a single threaded implementation of M-HASH and B-HASH on an Intel Xeon processor (2.86 GHz). Table 11 shows the average query response time. We note that M-HASH is substantially slower than B-HASH. This is because the number of collisions in the NAME_BUCKETING stage is higher for M-HASH.

We would like to point out that both NAME_BUCKETING and NAME_MATCHING stages can be multi-threaded on a multi-core machine and the query response time can be decreased by an order easily. Further, the memory footprint of the system is very small and the codewords require 4.1 MB for the employees name directory (150,000 names) and 13.8 MB for the Wikipedia name directory (550,000 names).

Test Set	MHASH	BHASH
DUMBTIONARY	190	87
INTRANET	148	75

Table 11: Average response time in milliseconds (single threaded system running on 2.86 GHz Intel Xeon Processor).

6 Related Work

Spelling correction of written text is a well studied problem (Kukich, 1996), (Jurafsky and Martin, 2008). The first approach to spelling correction made use of a lexicon to correct out-of-lexicon terms by finding the closest in-lexicon word (Damerau, 1964). The similarity between a misspelled word and an in-lexicon word was measured using Edit Distance (Jurafsky and Martin, 2008). The next class of approaches applied the noisy channel model to correct single word spelling errors (Kernighan et al., 1990), (Brill and Moore, 2000). A major flaw of single word spelling correction algorithms is they do not make use of the context of the word in correcting the errors. The next stream of approaches explored ways of exploiting the word's context (Golding and Roth, 1996), (Cucerzan and Brill, 2004). Recently, several works have leveraged the Web for improved spelling correction (Chen et al., 2007),(Islam and Inkpen, 2009), (Whitelaw et al., 2009). Spelling correction algorithms targeted for web-search queries have been developed making use of query logs and click-thru data (Cucerzan and Brill, 2004), (Ahmad and Kondrak, 2005), (Sun et al., 2010). None of these approaches focus exclusively on correcting name misspellings.

Name matching techniques have been studied in the context of database record deduplication, text mining, and information retrieval (Christen, 2006), (Pfeifer et al., 1996). Most techniques use one or more measures of phonetic similarity and/or string similarity. The popular phonetic similarity-based techniques are Soundex, Phonix, and Metaphone (Pfeifer et al., 1996). Some of the string similarity-based techniques employ Damerau-Levenshtein edit distance, Jaro distance or Winkler distance (Christen, 2006). Data driven approaches for learning edit distance have also been proposed (Ristad and Yianilos, 1996). Most of these techniques either give poor retrieval performance on large name directories or do not scale.

Hashing techniques for similarity search is also a well studied problem (Shakhnarovich et al., 2008). Locality Sensitive Hashing (LSH) is a theoretically grounded data-oblivious approach for using random projections to define the hash functions for data objects with a single view (Charikar, 2002), (Andoni and Indyk, 2006). Although LSH guarantees that asymptotically the Hamming distance between the codewords approaches the Euclidean distance between the data objects, it is known to produce long codewords making it practically inefficient. Recently data-aware approaches that employ Machine Learning techniques to learn hash functions have been proposed and shown to be a lot more effective than LSH on both synthetic and real data. Semantic Hashing employs Restricted Boltzmann Machine to produce more compact codes than LSH (Salakhutdinov and Hinton, 2009). Spectral Hashing formalizes the requirements for a good code and relates them to the problem of balanced graph partitioning which is known to be NP hard (Weiss et al., 2008). To give a practical algorithm for hashing, Spectral Hashing assumes that the data are sampled from a multidimensional uniform distribution and solves a relaxed partitioning problem.

7 Conclusions

We developed two hashing-based techniques for spelling correction of person names in People Search applications.To the best of our knowledge, these are the first techniques that focus exclusively on correcting spelling mistakes in person names. Our approach has several advantages over other spelling correction techniques. Firstly, we do not suggest incorrect suggestions for valid queries unlike (Cucerzan and Brill, 2004). Further, as we suggest spellings from only authoritative name directories, the suggestions are always well formed and coherent. Secondly, we do not require query logs and other resources that are not easily available unlike (Cucerzan and Brill, 2004), (Ahmad and Kondrak, 2005). Neither do we require pairs of misspelled names and their correct spellings for learning the error model unlike (Brill and Moore, 2000) or large-coverage general purpose lexicon for unlike (Cucerzan and Brill, 2004) or pronunciation dictionaries unlike (Toutanova and Moore, 2002). Thirdly, we correct the query as a whole unlike (Ahmad and Kondrak, 2005) and can handle word order changes unlike (Cucerzan and Brill, 2004). Fourthly, we do not iteratively process misspelled name unlike (Cucerzan and Brill, 2004). Fifthly, we handle large name directories efficiently unlike the spectrum of name matching techniques discussed in (Pfeifer et al., 1996). Finally, our training data requirement is relatively small.

As future work, we would like to explore the possibility of learning hash functions using 1) bilingual and monolingual data together and 2) multiple conjugate languages.

References

Farooq Ahmad and Grzegorz Kondrak. 2005. Learning a spelling error model from search query logs. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 955–962, Morristown, NJ, USA. Association for Computational Linguistics.

Alexandr Andoni and Piotr Indyk. 2006. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *FOCS*, pages 459–468.

Rahul Bhagat and Eduard H. Hovy. 2007. Phonetic models for generating spelling variants. In *IJCAI*, pages 1570–1575.

Mikhail Bilenko, Raymond J. Mooney, William W. Cohen, Pradeep D. Ravikumar, and Stephen E. Fienberg. 2003. Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23.

E. Brill and R. Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of ACL '00*, pages 286–293.

Moses Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388.

Qing Chen, Mu Li, and Ming Zhou. 2007. Improving query spelling correction using web search results. In *EMNLP-CoNLL*, pages 181–189.

P. Christen. 2006. A comparison of personal name matching: techniques and practical issues. *Technical Report TR-CS-06-02, Dept. of Computer Science, ANU, Canberra*.

William W. Cohen, Pradeep D. Ravikumar, and Stephen E. Fienberg. 2003. A comparison of string distance metrics for name-matching tasks. In *IIWeb*, pages 73–78.

S Cucerzan and E. Brill. 2004. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of EMNLP '04*, pages 293–300.

F.J. Damerau. 1964. A technique for computer detection and correction of spelling errors. *Communications of ACM*, 7(3):171–176.

C. Friedman and R. Sideli. 1992. Tolerating spelling errors during patient validation. *Computers and Biomedical Research*, 25:486–509.

Andrew R. Golding and Dan Roth. 1996. Applying winnow to context-sensitive spelling correction. *CoRR*, cmp-lg/9607024.

Gene H. Golub and Charles F. Van Loan. 1996. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 3rd edition.

David R. Hardoon, Sándor Szedmák, and John Shawe-Taylor. 2004. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16(12):2639–2664.

Aminul Islam and Diana Inkpen. 2009. Real-word spelling correction using google web 1tn-gram data set. In *CIKM*, pages 1689–1692.

D. Jurafsky and J.H. Martin. 2008. *Speech and Language Processing*. Prentice-Hall.

Mark D. Kernighan, Kenneth W. Church, and William A. Gale. 1990. A spelling correction program based on a noisy channel model. In *COLING*, pages 205–210.

K. Kukich. 1996. Techniques for automatically correcting words in a text. *Computing Surveys*, 24(4):377–439.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

G. Navarro, R. Baeza-Yates, and J. Azevedo-Arcoverde. 2003. Matchsimile: a flexible approximate matching tool for searching proper names. *Journal of the American Society for Information Science and Technology*, 54(1):3–15.

U. Pfeifer, T. Poersch, and N. Fuhr. 1996. Retrieval effectiveness of proper name search methods. *Information Processing and Management*, 32(6):667–679.

L. Philips. 2000. The double metaphone search algorithm. *C/C++ Users Journal*.

Eric Sven Ristad and Peter N. Yianilos. 1996. Learning string edit distance. *CoRR*, cmp-lg/9610005.

Ruslan Salakhutdinov and Geoffrey E. Hinton. 2009. Semantic hashing. *Int. J. Approx. Reasoning*, 50(7):969–978.

Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk. 2008. Nearest-neighbor methods in learning and vision. *IEEE Transactions on Neural Networks*, 19(2):377–377.

Xu Sun, Jianfeng Gao, Daniel Micol, and Chris Quirk. 2010. Learning phrase-based spelling error models from clickthrough data. In *Proceedings of ACL 2010*.

K. Toutanova and R. Moore. 2002. Pronounciation modeling for improved spelling correction. In *Proceedings of ACL '02*, pages 141–151.

Yair Weiss, Antonio B. Torralba, and Robert Fergus. 2008. Spectral hashing. In *NIPS*, pages 1753–1760.

Casey Whitelaw, Ben Hutchinson, Grace Chung, and Ged Ellis. 2009. Using the web for language independent spellchecking and autocorrection. In *EMNLP*, pages 890–899.

Identifying Functional Relations in Web Text

Thomas Lin, Mausam, Oren Etzioni
Turing Center
University of Washington
Seattle, WA 98195, USA
{tlin,mausam,etzioni}@cs.washington.edu

Abstract

Determining whether a textual phrase denotes a functional relation (*i.e.*, a relation that maps each domain element to a unique range element) is useful for numerous NLP tasks such as synonym resolution and contradiction detection. Previous work on this problem has relied on either counting methods or lexico-syntactic patterns. However, determining whether a relation is functional, by analyzing mentions of the relation in a corpus, is challenging due to ambiguity, synonymy, anaphora, and other linguistic phenomena.

We present the LEIBNIZ system that overcomes these challenges by exploiting the synergy between the Web corpus and freely-available knowledge resources such as Freebase. It first computes multiple *typed functionality* scores, representing functionality of the relation phrase when its arguments are constrained to specific types. It then aggregates these scores to predict the global functionality for the phrase. LEIBNIZ outperforms previous work, increasing area under the precision-recall curve from 0.61 to 0.88. We utilize LEIBNIZ to generate the first public repository of automatically-identified functional relations.

1 Introduction

The paradigm of Open Information Extraction (IE) (Banko et al., 2007; Banko and Etzioni, 2008) has scaled extraction technology to the massive set of relations expressed in Web text. However, additional work is needed to better understand these relations, and to place them in richer semantic structures. A step in that direction is identifying the properties of these relations, *e.g.*, symmetry, transitivity and our focus in this paper – functionality. We refer to this problem as *functionality identification*.

A binary relation is functional if, for a given arg1, there is exactly one unique value for arg2. Examples of functional relations are *father, death_date, birth_city*, *etc.* We define a *relation phrase* to be functional if all semantic relations commonly expressed by that phrase are functional. For example, we say that the phrase *'was born in'* denotes a functional relation, because the different semantic relations expressed by the phrase (*e.g.*, *birth_city, birth_year, etc.*) are all functional.

Knowing that a relation is functional is helpful for numerous NLP inference tasks. Previous work has used functionality for the tasks of contradiction detection (Ritter et al., 2008), quantifier scope disambiguation (Srinivasan and Yates, 2009), and synonym resolution (Yates and Etzioni, 2009). It could also aid in other tasks such as ontology generation and information extraction. For example, consider two sentences from a contradiction detection task:
(1) "George Washington *was born in* Virginia." and
(2) "George Washington *was born in* Texas."
As Ritter *et al.* (2008) points out, we can only determine that the two sentences are contradictory if we know that the semantic relation referred to by the phrase *'was born in'* is functional, and that both Virginia and Texas are distinct states.

Automatic functionality identification is essential when dealing with a large number of relations as in Open IE, or in complex domains where expert help

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1266–1276,
MIT, Massachusetts, USA, 9-11 October 2010.

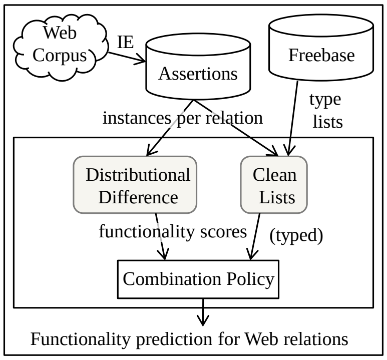


Figure 1: Our system, LEIBNIZ, uses the Web and Freebase to determine functionality of Web relations.

is scarce or expensive (*e.g.*, biomedical texts). This paper tackles automatic functionality identification using Web text. While functionality identification has been utilized as a module in various NLP systems, this is the first paper to focus exclusively on functionality identification as a *bona fide* NLP inference task.

It is natural to identify functions based on triples extracted from text instead of analyzing sentences directly. Thus, as our input, we utilize tuples extracted by TEXTRUNNER (Banko and Etzioni, 2008) when run over a corpus of 500 million webpages. TEXTRUNNER maps sentences to tuples of the form <arg1, *relation phrase*, arg2> and enables our LEIBNIZ system to focus on the problem of deciding whether the relation phrase is a function.

The naive approach, which classifies a relation phrase as non-functional if several arg1s have multiple arg2s in our extraction set, fails due to several reasons: synonymy – a unique entity may be referred by multiple strings, polysemy of both entities and relations – a unique string may refer to multiple entities/relations, metaphorical usage, extraction errors and more. These phenomena conspire to make the functionality determination task inherently statistical and surprisingly challenging.

In addition, a functional relation phrase may appear non-functional until we consider the types of its arguments. In our *'was born in'* example, <George Washington, *was born in*, 1732> does not contradict <George Washington, *was born in*, Virginia> even though we see two distinct arg2s for the same arg1. To solve functionality identification, we need to consider *typed relations* where the relations analyzed are constrained to have specific argument types.

We develop several approaches to overcome these challenges. Our first scheme employs approximate argument merging to overcome the synonymy and anaphora problems. Our second approach, DISTRDIFF, takes a statistical view of the problem and learns a separator for the typical count distributions of functional versus non-functional relations. Finally, our third and most successful scheme, CLEANLISTS, identifies and processes a cleaner subset of the data by intersecting the corpus with entities in a secondary knowledge-base (in our case, Freebase (Metaweb Technologies, 2009)). Utilizing pre-defined types, CLEANLISTS first identifies typed functionality for suitable types for that relation phrase, and then combines them to output a final functionality label. LEIBNIZ, a hybrid of CLEANLISTS and DISTRDIFF, returns state-of-the-art results for our task.

Our work makes the following contributions:

1. We identify several linguistic phenomena that make the problem of corpus-based functionality identification surprisingly difficult.

2. We designed and implemented three novel techniques for identifying functionality based on instance-based counting, distributional differences, and use of external knowledge bases.

3. Our best method, LEIBNIZ, outperforms the existing approaches by wide margins, increasing area under the precision-recall curve from 0.61 to 0.88. It is also capable of distinguishing functionality of typed relation phrases, when the arguments are restricted to specific types.

4. Utilizing LEIBNIZ, we created the first public repository of functional relations.[1]

2 Related Work

There is a recent surge in large knowledge bases constructed by human collaboration such as Freebase (Metaweb Technologies, 2009) and VerbNet (Kipper-Schuler, 2005). VerbNet annotates its verbs with several properties but not functionality. Freebase does annotate some relations with an 'is unique' property, which is similar to functionality, but the number of relations in Freebase is still much

[1] available at http://www.cs.washington.edu/research/leibniz

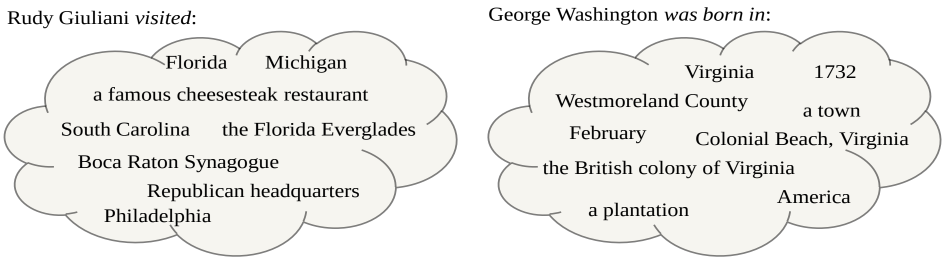


Figure 2: Sample arg2 values for a non-functional relation (*visited*) *vs.* a functional relation (*was born in*) illustrate the challenge in discriminating functionality from Web text.

smaller than the hundreds of thousands of relations existing on the Web, necessitating automatic approaches to functionality identification.

Discovering functional dependencies has been recognized as an important database analysis technique (Huhtala et al., 1999; Yao and Hamilton, 2008), but the database community does not address any of the linguistic phenomena which make this a challenging problem in NLP. Three groups of researchers have studied functionality identification in the context of natural language.

AuContraire (Ritter et al., 2008) is a contradiction detection system that also learns relation functionality. Their approach combines a probabilistic model based on (Downey et al., 2005) with estimates on whether each arg1 is ambiguous. The estimates are used to weight each arg1's contribution to an overall functionality score for each relation. Both argument-ambiguity and relation-functionality are jointly estimated using an EM-like method. While elegant, AuContraire requires substantial hand-engineered knowledge, which limits the scalability of their approach.

Lexico-syntactic patterns: Srinivasan and Yates (2009) disambiguate a quantifier's scope by first making judgments about relation functionality. For functionality, they look for *numeric phrases* following the relation. For example, the presence of the numeric term 'four' in the sentence "the fire *destroyed* four shops" suggests that *destroyed* is not functional, since the same arg1 can destroy multiple things.

The key problem with this approach is that it often assigns different functionality labels for the present tense and past tense phrases of the same semantic relation. For example, it will consider *'lived in'* to be non-functional, but *'lives in'* to be functional, since we rarely say "someone lives in many cities". Since both these phrases refer to the same semantic relation this approach has low precision. Moreover, it performs poorly for relation phrases that naturally expect numbers as the target argument (*e.g.*, *'has an atomic number of'*).

While these lexico-syntactic patterns do not perform as well for our task, they are well-suited for identifying whether a verb phrase can take multiple objects or not. This can be understood as a functionality property of the verb phrase within a sentence, as opposed to functionality of the semantic relation the phrase represents.

WIE: In a preliminary study, Popescu (2007) applies an instance based counting approach, but her relations require manually annotated type restrictions, which makes the approach less scalable.

Finally, functionality is just one property of relations that can be learned from text. A number of other studies (Guarino and Welty, 2004; Volker et al., 2005; Culotta et al., 2006) have examined detecting other relation properties from text and applying them to tasks such as ontology cleaning.

3 Challenges for Functionality Identification

A functional binary relation r is formally defined as one such that $\forall x, y_1, y_2 : r(x, y_1) \wedge r(x, y_2) \Rightarrow y_1 = y_2$. We define a relation string to be functional if all semantic relations commonly expressed by the relation string are individually functional. Thus, under our definition, *'was born in'* and *'died in'* are functional, even though they can take different arg2s for the same arg1, *e.g.*, year, city, state, country, *etc.*

The definition of a functional relation suggests a naive instance-based counting algorithm for identifying functionality. "Look for the number of arg2s for each arg1. If all (or most) arg1s have exactly one arg2, label the relation phrase functional, else, non-functional." Unfortunately, this naive algorithm fails for our task exposing several linguistic phenomena that make our problem hard (see Figure 2):

Synonymy: Various arg2s for the same arg1 may refer to the same entity. This makes many functional relations seem non-functional. For instance, <George Washington, *was born in*, Virginia> and <George Washington, *was born in*, the British colony of Virginia> are not in conflict. Other examples of synonyms include 'Windy City' and 'Chicago'; '3rd March' and '03/03', *etc.*

Anaphora: An entity can be referred to by using several phrases. For instance, <George Washington, *was born in*, a town> does not conflict with his being born in 'Colonial Beach, Virginia', since 'town' is an anaphora for his city of birth. Other examples include 'The US President' for 'George W. Bush', and 'the superpower' to refer to 'United States'. The effect is similar to that of synonyms – many relations incorrectly appear non-functional.

Argument Ambiguity: <George Washington, *was born in*, 'Kortrijk, Belgium'> in addition to his being born in 'Virginia' suggests that *'was born in'* is non-functional. However, the real cause is that 'George Washington' is ambiguous and refers to different people. This ambiguity gets more pronounced if the person is referred to just by their first (or last name), *e.g.*, 'Clinton' is commonly used to refer to both Hillary and Bill Clinton.

Relation Phrase Ambiguity: A relation phrase can have several senses. For instance '*weighs* 80 kilos' is a different *weighs* than '*weighs* his options'.

Type Restrictions: A closely related problem is type-variations in the argument. *E.g.*, <George Washington, *was born in*, America> *vs.* <George Washington, *born in*, Virginia> both use the same sense of *'was born in'* but refer to different semantic relations – one that takes a country in arg2, and the other that takes a state. Moreover, different argument types may result in different functionality labels. For example, *'published in'* is functional if the arg2 is a year, but non-functional if it is a language, since a book could be published in many languages. We refer to this finer notion of functionality as *typed functionality*.

Data Sparsity: There is limited data for more obscure relations instances and non-functional relation phrases appear functional due to lack of evidence.

Textually Functional Relations: Last but not least, some relations that are not functional may appear functional in text. An example is *'collects'*. We collect many things, but rarely mention it in text. Usually, someone's collection is mentioned in text only when it makes the news. We name such relations *textually functional*. Even though we could build techniques to reduce the impact of other phenomena, no instance based counting scheme could overcome the challenge posed by textually functional relations.

Finally, we note that our functionality predictor operates over tuples generated by an Open IE system. The extractors are not perfect and their errors can also complicate our analysis.

4 Algorithms

To overcome these challenges, we design three algorithms. Our first algorithm, IBC, applies several rules to determine whether two arg2s are equal. Our second algorithm, DISTRDIFF, takes a statistical approach, and tries to learn a discriminator between typical count distributions for functional and non-functional relations. Our final approach, CLEANLISTS, applies counting over a cleaner subset of the corpus, which is generated based on entities present in a secondary KB such as Freebase.

From this section onwards, we gloss over the distinction between a semantic relation and a relation phrase, since our algorithms do not have access to relations and operate only at the phrase level. We use 'relation' to refer to the phrases.

4.1 Instance Based Counting (IBC)

For each relation, IBC computes a global functionality score by aggregating local functionality scores for each arg1. The local functionality for each arg1 computes the fraction of arg2 pairs that refer to the same entity. To operationalize this computation we need to identify which arg2s co-refer. Moreover, we also need to pick an aggregation strategy to combine local functionality scores.

Data Cleaning: Common nouns in arg1s are often anaphoras for other entities. For example, <the company, *was headquartered in*, ...> refers to different companies in different extractions. To combat this, IBC restricts arg1s to proper nouns. Secondly, to counter extraction errors and data bias, it retains

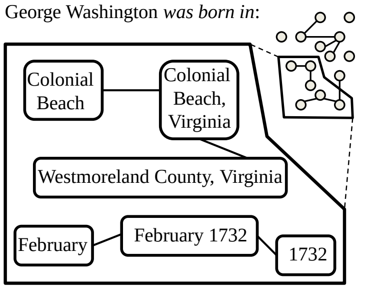


Figure 3: IBC judges that *Colonial Beach* and *Westmoreland County, Virginia* refer to the same entity.

an extraction only once per unique sentence. This reduces the disproportionately large frequencies of some assertions that are generated from a single article published at multiple websites. Similarly, it allows an extraction only once per website url. Moreover, it filters out any arg1 that does not appear at least 10 times with that relation.

Equality Checking: This key component judges if two arg2s refer to the same entity. It first employs weak typing by disallowing equality checks across common nouns, proper nouns, dates and numbers. This mitigates the relation ambiguity problem, since we never compare *'born in*(1732)' and *'born in*(Virginia)'. Within the same category it judges two arg2s to co-refer if they share a content word. It also performs a connected component analysis (Hopcraft and Tarjan, 1973) to take a transitive closure of arg2s judged equal (see Figure 3).

For example, for the relation *'was named after'* and arg1='Bluetooth' our corpus has three arg2s: 'Harald Bluetooth', 'Harald Bluetooth, the King of Denmark' and 'the King of Denmark'. Our equality method judges all three as referring to the same entity. Note that this is a heuristic approach, which could make mistakes. But for an error, there needs to be extractions with the same arg1, relation and similar arg2s. Such cases exist, but are not common. Our equality checking mitigates the problems of anaphora, synonymy as well as some typing.

Aggregation: We try several methods to aggregate local functionality scores for each arg1 into a global score for the relation. These include, a simple average, a weighted average weighted by frequency of each arg1, a weighted average weighted by log of frequency of each arg1, and a Bayesian approach that estimates the probability that a relation is functional using statistics over a small development set. Overall, the log-weighting works the best: it assigns a higher score for popular arguments, but not so high that it drowns out all the other evidence.

4.2 DISTRDIFF

Our second algorithm, DISTRDIFF, takes a purely statistical, discriminative view of the problem. It recognizes that, due to aforementioned reasons, whether a relation is functional or not, there are bound to be several arg1s that look locally functional and several that look locally non-functional. The difference is in the *number* of such arg1s – a functional relation will have more of the former type.

DISTRDIFF studies the count distributions for a small development set of functional relations (and similarly for non-functional) and attempts to build a separator between the two. As an illustration, Figure 4(a) plots the arg2 counts for various arg1s for a functional relation (*'is headquartered in'*). Each curve represents a unique arg1. For an arg1, the x-axis represents the rank (based on frequency) of arg2s and y-axis represents the normalized frequency of the arg2. For example, if an arg1 is found with just one arg2, then x=1 will match with y=1 (the first point has all the mass) and x=2 will match with y=0. If, on the other hand, an arg1 is found with five arg2s, say, appearing ten times each, then the first five x-points will map to 0.2 and the sixth point will map to 0.

We illustrate the same plot for a non-functional relation (*'visited'*) in Figure 4(b). It is evident from the two figures that, as one would expect, curves for most arg1s die early in case of a functional relation, whereas the lower ranked arg2s are more densely populated in case of a non-functional relation.

We aggregate this information using *slope* of the best-fit line for each arg1 curve. For functional relations, the best-fit lines have steep slopes, whereas for non-functional the lines are flatter. We bucket the slopes in integer bins and count the fraction of arg1s appearing in each bin. This lets us aggregate the information into a single slope-distribution for each relation. Bold lines in Figure 4(c) illustrate the average slope-distributions, averaged over ten sample relations of each kind – dashed for non-functional and solid for functional. Most non-functional relations have a much higher probability of arg1s with low magnitude slopes, whereas functional relations are

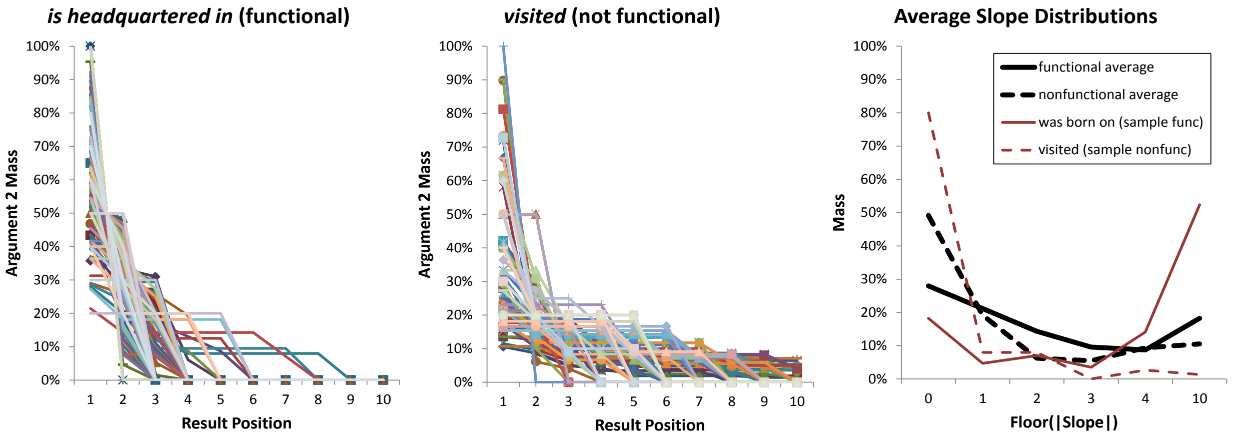


Figure 4: DISTRDIFF: Arg2 count distributions fall more sharply for (a) a sample functional relation, than (b) a sample non-functional relation. (c) The distance of aggregated slope-distributions from average slope-distributions can be used to predict the functionality.

the opposite. Notice that the aggregated curve for *'visited'* in the figure is closer to the average curve for non-functional than to functional and vice-versa for *'was born on'*.

We plot the aggregated slope-distributions for each relation and use the distance from average distributions as a means to predict the functionality. We use KL divergence (Kullback and Leibler, 1951) to compute the distance between two distributions. We score a relation's functionality in three ways using: (1) KLFUNC, its distance from average functional slope-distribution F_{avg}, (2) KLDIFF, its distance from average functional minus its distance from average non-functional N_{avg}, and (3) average of these two scores. For a relation with slope distribution R, the scores are computed as:

$$\text{KLFUNC} = \sum_i R(i) ln \frac{R(i)}{F_{avg}(i)}$$
$$\text{KLDIFF} = \text{KLFUNC} - (\sum_i R(i) ln \frac{R(i)}{N_{avg}(i)})$$

Section 5.2 compares the three scoring functions. A purely statistical approach is resilient to noisy data, and does not need to explicitly account for the various issues we detailed earlier. A disadvantage is that it cannot handle relation ambiguity and type restrictions. Moreover, we may need to relearn the separator if applying DISTRDIFF to a corpus with very different count distributions.

4.3 CLEANLISTS

Our third algorithm, CLEANLISTS, is based on the intuition that for identifying functionality we need not reason over all the data in our corpus; instead, a small but cleaner subset of the data may work best. This clean subset should ideally be free of synonyms, ambiguities and anaphora, and be typed.

Several knowledge-bases such as Wordnet, Wikipedia, and Freebase (Fellbaum, 1998; Wikipedia, 2004; Metaweb Technologies, 2009), are readily and freely available and they all provide clean typed lists of entities. In our experiments CLEANLISTS employs Freebase as a source of clean lists, but we could use any of these or other domain-specific ontologies such as SNOMED (Price and Spackman, 2000) as well.

CLEANLISTS takes the intersection of Freebase entities with our corpus to generate a clean subset for functionality analysis. Freebase currently has over 12 million entities in over 1,000 typed lists. Thus, this intersection retains significant portions of the useful data, and gets rid of most of anaphora and synonymy issues. Moreover, by matching against typed lists, many relation ambiguities are separated as well, since ambiguous relations often take different types in the arguments (*e.g.*, *'ran*(Distance)' *vs.* *'ran*(Company)'). To mitigate the effect of argument ambiguity, we additionally get rid of instances in which arg1s match multiple names in the Freebase list of names.

As an example, consider the *'was born in'* relation. CLEANLISTS will remove instances with only 'Clinton' in arg1, since it matches multiple people in Freebase. It will treat the different types, *e.g.*, cities, states, countries, months separately and analyze the functionality for each of these individually.

By intersecting the relation data with argument lists for these types, we will be left with a smaller, but much cleaner, subset of relation data, one for each type. CLEANLISTS analyzes each subset using simple, instance based counting and computes a typed functionality score for each type. Thus, it first computes typed functionality for each relation.

There are two subtleties in applying this algorithm. First, we need to identify the set of types to consider for each relation. Our algorithm currently picks the types that occur most in each relation's observed data. In the future, we could also use a selectional preferences system (Ritter et al., 2010; Kozareva and Hovy, 2010). Note that we remove Freebase types such as *Written Work* from consideration for containing many entities whose primary senses are not that type. For example, both 'Al Gore' and 'William Clinton' are also names of books, but references in text to these are rarely a reference to the written work sense.

Secondly, an argument could belong to multiple Freebase lists. For example, 'California' is both a city and a state. We apply a simple heuristic: if a string appears in multiple lists under consideration, we assign it to the smallest of the lists (the list of cities is much larger than states). This simple heuristic usually assigns an argument to its intended type. On a development set, the error rate of this heuristic is <4%, though it varies a bit depending on the types involved.

CLEANLISTS determines the overall functionality of a relation string by aggregating the scores for each type. It outputs functional if a majority of typed senses for the relation are functional. For example, CLEANLISTS judges *'was born in'* to be functional, since all relevant type restrictions are individually typed functional – everyone is born in exactly one country, city, state, month, *etc.*

CLEANLISTS has a much higher precision due to the intersection with clean lists, though at some cost of recall. The reason for lower recall is that the approach has a bias towards types that are easy to enumerate. It does not have different distances (*e.g.*, 50 kms, 20 miles, *etc.*) in its lists. Moreover, arguments that do not correspond to a noun cannot be handled. For example, in the sentence, "He *weighed* eating a cheeseburger against eating a salad", the arg2 of *'weighed'* can't be matched to a Freebase list. To increase the recall we back off to DISTRDIFF in the cases when CLEANLISTS is unable to make a prediction. This combination gives the best balance of precision and recall for our task. We name our final system LEIBNIZ.

One current limitation is that using only those arg2s that exactly match clean lists leaves out some good data (*e.g.*, a tuple with an arg2 of 'Univ of Wash' will not match against a list of universities that spells it as 'University of Washington'). Because we have access to entity types, using typed equality checkers (Prager et al., 2007) with the clean lists would allow us to recapture much of this useful data. Moreover, the knowledge of functions could apply to building new type nanotheories and reduce considerable manual effort. We wish to study this in the future.

5 Evaluation

In our evaluation, we wish to answer three questions: (1) How do our three approaches, *Instance Based Counting* (IBC), DISTRDIFF, and CLEANLISTS, compare on the functionality identification task? (2) How does our final system, LEIBNIZ, compare against the existing state of the art techniques? (3) How well is LEIBNIZ able to identify typed functionality for different types in the same relation phrase?

5.1 Dataset

For our experiments we test on the set of 887 relations used by Ritter *et al.* (2008) in their experiments. We use the Open IE corpus generated by running TEXTRUNNER on 500 million high quality Webpages (Banko and Etzioni, 2008) as the source of instance data for these relations. Extractor and corpus differences lead to some relations not occurring (or not occurring with sufficient frequency to properly analyze, *i.e.*, ≥ 5 arg1 with ≥ 10 evidence), leaving a dataset of 629 relations on which to test.

Two human experts tagged these relations for functionality. Tagging the functionality of relation phrases can be a bit subjective, as it requires the experts to imagine the various senses of a phrase and judge functionality over all those senses. The inter-annotator agreement between the experts was 95.5%. We limit ourselves to the subset of the data

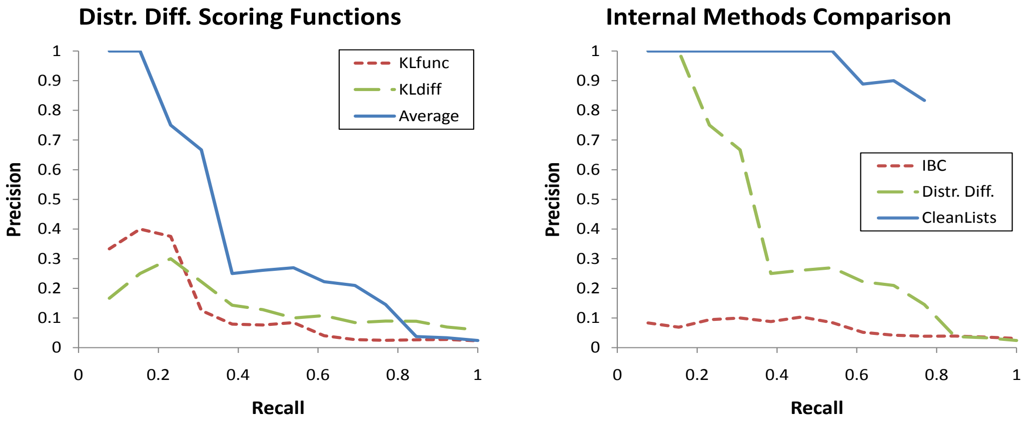


Figure 5: (a) The best scoring method for DistrDiff averages KLfunc and KLdiff. (b) CleanLists performs significantly better than DistrDiff, which performs significantly better than IBC.

on which the two experts agreed (a subset of 601 relation phrases).

5.2 Internal Comparisons

First, we compare the three scoring functions for DistrDiff. We vary the score thresholds to generate the different points on the precision-recall curves for each of the three. Figure 5(a) plots these curves. It is evident that the hybrid scoring function, *i.e.*, one which is an average of KLfunc (distance from average functional) and KLdiff (distance from average functional minus distance from average non-functional) performs the best. We use this scoring in the further experiments involving DistrDiff.

Next, we compare our three algorithms on the dataset. Figure 5(b) reports the results. CleanLists outperforms DistrDiff by vast margins, covering a 33.5% additional area under the precision-recall curve. Overall, CleanLists finds the very high precision points, because of its use of clean data. However, it is unable to make 23.1% of the predictions, primarily because the intersection between the corpus and Freebase entities results in very little data for those relations. DistrDiff performs better than IBC, due to its statistical nature, but the issues described in Section 3 plague both these systems much more than CleanLists.

To increase the recall Leibniz uses a combination of DistrDiff and CleanLists, in which the algorithm backs off to DistrDiff if CleanLists is unable to output a prediction.

5.3 External Comparisons

We next compare Leibniz against the existing state of the art approaches. Our competitors are AuContraire and NumericTerms (Ritter et al., 2008; Srinivasan and Yates, 2009). Because we use the AuContraire dataset, we report the results from their best performing system. We reimplement a version of NumericTerms using their list of numeric quantifiers and extraction patterns that best correspond to our relation format. We run our implementation of NumericTerms on a dataset of 100 million English sentences from a crawl of high quality Webpages to generate the functionality labels.

Figure 6(a) reports the results of this experiment. We find that Leibniz outperforms AuContraire by vast margins covering an additional 44% area in the precision-recall curve. AuContraire's AUC is 0.61 whereas Leibniz covers 0.88. A Bootstrap Percentile Test (Keller et al., 2005) on F_1 score found the improvement of our techniques over AuContraire to be statistically significant at $\alpha = 0.05$. NumericTerms does not perform well, because it makes decisions based only on the local evidence in a sentence, and does not integrate the knowledge from different occurrences of the same relation. It returns many false positives, such as *'lives in'*, which appear functional to the lexico-syntactic pattern, but are clearly non-functional, *e.g.*, one could live in many places over a lifetime.

An example of a Leibniz error is the *'represented'* relation. Leibniz classifies this as functional, because it finds several strongly functional senses (*e.g.*, when a person represents a country),

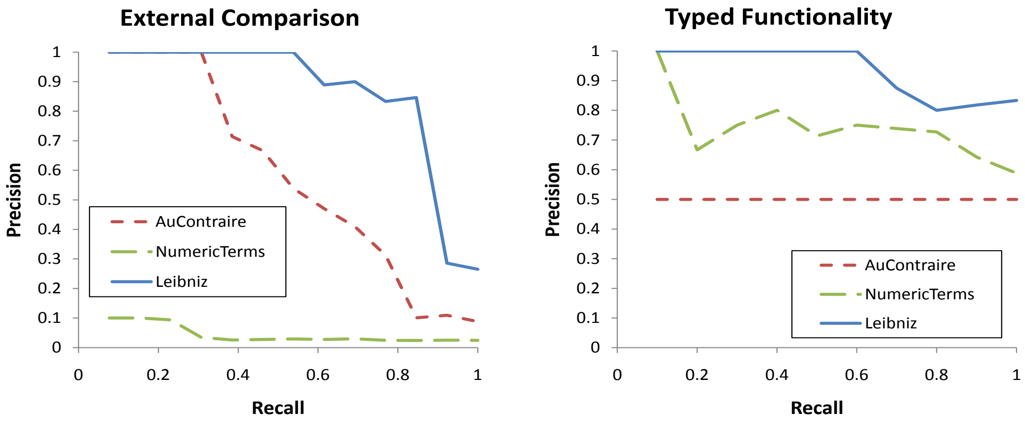


Figure 6: (a) LEIBNIZ, which is a hybrid of CLEANLISTS and DISTRDIFF, achieves 0.88 AUC and outperforms the 0.61 AUC from AuContraire (Ritter et al., 2008) and the 0.05 AUC from NumericTerms (Srinivasan and Yates, 2009). (b) LEIBNIZ is able to tease apart different senses of polysemous relations much better than other systems.

but the human experts might have had some non-functional senses in mind while labeling.

5.4 Typed Functionality

Next, we conduct a study of the typed functionality task. We test on ten common polysemous relations, each having both a functional and a non-functional sense. An example is the *'was published in'* relation. If arg2 is a year it is functional, *e.g.* <Harry Potter 5, *was published in*, 2003>. However, *'was published in*(Language)' is not functional, *e.g.* <Harry Potter 5, *was published in*, [French / Spanish / English]>. Similarly, *'will become*(Company)' is functional because when a company is renamed, it transitions away from the old name exactly once, *e.g.* <Cingular, *will become*, AT&T Wireless>. However, *'will become*(government title)' is not functional, because people can hold different offices in their life, *e.g.*, <Obama, *will become*, [Senator / President]>.

In this experiment, a simple baseline of predicting the same label for the two types of each relation achieves a precision of 0.5. Figure 6(b) presents the results of this study. AuContraire achieves a flat 0.5, since it cannot distinguish between types. NumericTerms can be modified to distinguish between basic types – check the word just after the numeric term to see whether it matches the type name. For example, the modified NumericTerms will search the Web for instances of *"was published in* [numeric term] *years" vs. "was published in* [numeric term] *languages"*. This scheme works better when the type name is simple (*e.g.*, languages) rather than complex (*e.g.*, government titles).

LEIBNIZ performs the best and is able to tease apart the functionality of various types very well. When LEIBNIZ did not work, it was generally because of textual functionality, which is a larger issue for typed functionality than general functionality. Of course, these results are merely suggestive – we perform a larger-scale experiment and generate a repository of typed functions next.

6 A Repository of Functional Relations

We now report on a repository of typed functional relations generated automatically by applying LEIBNIZ to a large collection of relation phrases. Instead of starting with the most frequent relations from TEXTRUNNER, we use OCCAM's relations (Fader et al., 2010) because they are more specific. For instance, where TEXTRUNNER outputs an underspecified tuple, <Gold, *has*, an atomic number of 79>, OCCAM extracts <Gold, *has an atomic number of*, 79>. OCCAM enables LEIBNIZ to identify far more functional relations than TEXTRUNNER.

6.1 Addressing Evidence Sparsity

Scaling up to a large collection of typed relations requires us to consider the size of our data sets. For example, consider which relation is more likely to be functional—a relation with 10 instances all of which indicate functionality versus a relation with 100 instances where 95 behave functionally.

To address this problem, we adapt the likelihood ratio approach from Schoenmackers *et al.* (2010).

For a typed relation with *n* instances, *f* of which indicate functionality, the G-test (Dunning, 1993), $G = 2*(f*\ln(f/k)+(n-f)*\ln((n-f)/(n-k)))$, provides a measure for the likelihood that the relation is not functional. Here *k* denotes the evidence indicating functionality for the case where the relation is not functional. Setting $k = n*0.25$ worked well for us. This G-score replaces our previous metric for scoring functional relations.

6.2 Evaluation of the Repository

In CLEANLISTS a factor that affects the quality of the results is the exact set of lists that is used. If the lists are not clean, results get noisy. For example, Freebase's list of *films* contains 73,000 entries, many of which (*e.g.*, "*Egg*") are not films in their primary senses. Even with heuristics such as assigning terms to their smallest lists and disqualifying dictionary words that occur from large type lists, there is still significant noise left.

Using LEIBNIZ with a set of 35 clean lists on OCCAM's extraction corpus, we generated a repository of 5,520 typed functional relations. To evaluate this resource a human expert tagged a random subset of the top 1,000 relations. Of these relations 22% were either ill-formed or had non-sensical type constraints. From the well-formed typed relations the precision was estimated to be 0.8. About half the errors were due to textual functionality and the rest were LEIBNIZ errors. Some examples of good functions found include *isTheSequelTo(videogame)* and *areTheBirthstoneFor(month)*. An example of a textually functional relation found is *wasTheFounderOf(company).*

This is the first public repository of automatically-identified functional relations. Scaling up our data set forced us to confront new sources of noise including extractor errors, errors due to mismatched types, and errors due to sparse evidence. Still, our initial results are encouraging and we hope that our resource will be valuable as a baseline for future work.

7 Conclusions

Functionality identification is an important subtask for Web-scale information extraction and other machine reading tasks. We study the problem of predicting the functionality of a relation phrase automatically from Web text. We presented three algorithms for this task: (1) instance-based counting, (2) DISTRDIFF, which takes a statistical approach and discriminatively classifies the relations using average arg-distributions, and (3) CLEANLISTS, which performs instance based counting on a subset of clean data generated by intersection of the corpus with a knowledge-base like Freebase.

Our best approach, LEIBNIZ, is a hybrid of DISTRDIFF and CLEANLISTS, and outperforms the existing state-of-the-art approaches by covering 44% more area under the precision-recall curve. We also observe that an important sub-component of identifying a functional relation phrase is identifying typed functionality, *i.e.*, functionality when the arguments of the relation phrase are type-constrained. Because CLEANLISTS is able to use typed lists, it can successfully identify typed functionality.

We run our techniques on a large set of relations to output a first repository of typed functional relations. We release this list for further use by the research community.[2]

Future Work: Functionality is one of the several properties a relation can possess. Others include selectional preferences, transitivity (Schoenmackers et al., 2008), mutual exclusion, symmetry, *etc.* These properties are very useful in increasing our understanding about these Open IE relation strings. We believe that the general principles developed in this work, for example, connecting the Open IE knowledge with an existing knowledge resource, will come in very handy in identifying these other properties.

Acknowledgements

We would like to thank Alan Ritter, Alex Yates and Anthony Fader for access to their data sets. We would like to thank Stephen Soderland, Yoav Artzi, and the anonymous reviewers for helpful comments on previous drafts. This research was supported in part by NSF grant IIS-0803481, ONR grant N00014-08-1-0431, DARPA contract FA8750-09-C-0179, a NDSEG Fellowship, and carried out at the University of Washington's Turing Center.

[2] available at `http://www.cs.washington.edu/research/leibniz`

References

M. Banko and O. Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*.

M. Banko, M. 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)*.

A. Culotta, A. McCallum, and J. Betz. 2006. Integrating probabilistic extraction models and data mining to discover relations and patterns in text. In *Proceedings of the HLT-NAACL*.

D. Downey, O. Etzioni, and S. Soderland. 2005. A probabilistic model of redundancy in information extraction. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*.

T. Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. In *Computational Linguistics*, volume 19.

A. Fader, O. Etzioni, and S. Soderland. 2010. Identifying well-specified relations for open information extraction. (In preparation).

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

N. Guarino and C. Welty. 2004. An overview of OntoClean. In *Handbook of Ontologies in Information Systems*, pages 151–172.

J. Hopcraft and R. Tarjan. 1973. Efficient algorithms for graph manipulation. *Communications of the ACM*, 16:372–378.

Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen. 1999. TANE: An efficient algorithm for discovering functional and approximate dependencies. In *The Computer Journal*.

M. Keller, S. Bengio, and S.Y. Wong. 2005. Benchmarking non-parametric statistical tests. In *Advances in Neural Information Processing Systems (NIPS) 18*.

K. Kipper-Schuler. 2005. Verbnet: A broad-coverage, comprehensive verb lexicon. In *Ph.D. thesis. University of Pennsylvania*.

Z. Kozareva and E. Hovy. 2010. Learning arguments and supertypes of semantic relations using recursive patterns. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*.

S. Kullback and R.A. Leibler. 1951. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86.

Metaweb Technologies. 2009. Freebase data dumps. In *http://download.freebase.com/datadumps/*.

A-M. Popescu. 2007. Information extraction from unstructured web text. In *Ph.D. thesis. University of Washington*.

J. Prager, S. Luger, and J. Chu-Carroll. 2007. Type nanotheories: a framework for term comparison. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management (CIKM)*.

C. Price and K. Spackman. 2000. SNOMED clincal terms. In *British Journal of Healthcare Computing & Information Management*, volume 17.

A. Ritter, D. Downey, S. Soderland, and O. Etzioni. 2008. It's a contradiction - no, it's not: A case study using functional relations. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

A. Ritter, Mausam, and O. Etzioni. 2010. A latent dirichlet allocation method for selectional preferences. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*.

S. Schoenmackers, O. Etzioni, and D. Weld. 2008. Scaling textual inference to the web. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

S. Schoenmackers, J. Davis, O. Etzioni, and D. Weld. 2010. Learning first-order horn clauses from web text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

P. Srinivasan and A. Yates. 2009. Quantifier scope disambiguation using extracted pragmatic knowledge: Preliminary results. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

J. Volker, D. Vrandecic, and Y. Sure. 2005. Automatic evaluation of ontologies (AEON). In *Proceedings of the 4th International Semantic Web Conference (ISWC)*.

Wikipedia. 2004. Wikipedia: The free encyclopedia. In *http://www.wikipedia.org*. Wikimedia Foundation.

H. Yao and H. Hamilton. 2008. Mining functional dependencies from data. In *Data Mining and Knowledge Discovery*.

A. Yates and O. Etzioni. 2009. Unsupervised methods for determining object and relation synonyms on the web. In *Journal of Artificial Intelligence Research*, volume 34, pages 255–296.

A Latent Variable Model for Geographic Lexical Variation

Jacob Eisenstein Brendan O'Connor Noah A. Smith Eric P. Xing
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{jacobeis,brendano,nasmith,epxing}@cs.cmu.edu

Abstract

The rapid growth of geotagged social media raises new computational possibilities for investigating geographic linguistic variation. In this paper, we present a multi-level generative model that reasons jointly about latent topics and geographical regions. High-level topics such as "sports" or "entertainment" are rendered differently in each geographic region, revealing topic-specific regional distinctions. Applied to a new dataset of geotagged microblogs, our model recovers coherent topics and their regional variants, while identifying geographic areas of linguistic consistency. The model also enables prediction of an author's geographic location from raw text, outperforming both text regression and supervised topic models.

1 Introduction

Sociolinguistics and dialectology study how language varies across social and regional contexts. Quantitative research in these fields generally proceeds by counting the frequency of a handful of previously-identified linguistic *variables*: pairs of phonological, lexical, or morphosyntactic features that are semantically equivalent, but whose frequency depends on social, geographical, or other factors (Paolillo, 2002; Chambers, 2009). It is left to the experimenter to determine which variables will be considered, and there is no obvious procedure for drawing inferences from the distribution of multiple variables. In this paper, we present a method for identifying geographically-aligned lexical variation directly from raw text. Our approach takes the form of a probabilistic graphical model capable of identifying both geographically-salient terms and coherent linguistic communities.

One challenge in the study of lexical variation is that term frequencies are influenced by a variety of factors, such as the topic of discourse. We address this issue by adding latent variables that allow us to model topical variation explicitly. We hypothesize that geography and topic interact, as "pure" topical lexical distributions are corrupted by geographical factors; for example, a sports-related topic will be rendered differently in New York and California. Each author is imbued with a latent "region" indicator, which both selects the regional variant of each topic, and generates the author's observed geographical location. The regional corruption of topics is modeled through a cascade of logistic normal priors—a general modeling approach which we call *cascading* topic models. The resulting system has multiple capabilities, including: (i) analyzing lexical variation by both topic and geography; (ii) segmenting geographical space into coherent linguistic communities; (iii) predicting author location based on text alone.

This research is only possible due to the rapid growth of social media. Our dataset is derived from the microblogging website Twitter,[1] which permits users to post short messages to the public. Many users of Twitter also supply exact geographical coordinates from GPS-enabled devices (e.g., mobile phones),[2] yielding *geotagged* text data. Text in computer-mediated communication is often more vernacular (Tagliamonte and Denis, 2008), and as such it is more likely to reveal the influence of geographic factors than text written in a more formal genre, such as news text (Labov, 1966).

We evaluate our approach both qualitatively and quantitatively. We investigate the topics and regions

[1] http://www.twitter.com

[2] User profiles also contain self-reported location names, but we do not use that information in this work.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1277–1287,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

that the model obtains, showing both common-sense results (place names and sports teams are grouped appropriately), as well as less-obvious insights about slang. Quantitatively, we apply our model to predict the location of unlabeled authors, using text alone. On this task, our model outperforms several alternatives, including both discriminative text regression and related latent-variable approaches.

2 Data

The main dataset in this research is gathered from the microblog website Twitter, via its official API. We use an archive of messages collected over the first week of March 2010 from the "Gardenhose" sample stream,[3] which then consisted of 15% of all public messages, totaling millions per day. We aggressively filter this stream, using only messages that are tagged with physical (latitude, longitude) coordinate pairs from a mobile client, and whose authors wrote at least 20 messages over this period. We also filter to include only authors who follow fewer than 1,000 other people, and have fewer than 1,000 followers. Kwak et al. (2010) find dramatic shifts in behavior among users with social graph connectivity outside of that range; such users may be marketers, celebrities with professional publicists, news media sources, etc. We also remove messages containing URLs to eliminate bots posting information such as advertising or weather conditions. For interpretability, we restrict our attention to authors inside a bounding box around the contiguous U.S. states, yielding a final sample of about 9,500 users and 380,000 messages, totaling 4.7 million word tokens. We have made this dataset available online.[4]

Informal text from mobile phones is challenging to tokenize; we adapt a publicly available tokenizer[5] originally developed for Twitter (O'Connor et al., 2010), which preserves emoticons and blocks of punctuation and other symbols as tokens. For each user's Twitter feed, we combine all messages into a single "document." We remove word types that appear in fewer than 40 feeds, yielding a vocabulary of 5,216 words. Of these, 1,332 do not appear in the English, French, or Spanish dictionaries of the spell-checking program `aspell`.

Every message is tagged with a location, but most messages from a single individual tend to come from nearby locations (as they go about their day); for modeling purposes we use only a single geographic location for each author, simply taking the location of the first message in the sample.

The authors in our dataset are fairly heavy Twitter users, posting an average of 40 messages per day (although we see only 15% of this total). We have little information about their demographics, though from the text it seems likely that this user set skews towards teens and young adults. The dataset covers each of the 48 contiguous United States and the District of Columbia.

3 Model

We develop a model that incorporates two sources of lexical variation: topic and geographical region. We treat the text and geographic locations as outputs from a generative process that incorporates both topics and regions as latent variables.[6] During inference, we seek to recover the topics and regions that best explain the observed data.

At the base level of model are "pure" topics (such as "**sports**", "**weather**", or "**slang**"); these topics are rendered differently in each region. We call this general modeling approach a *cascading* topic model; we describe it first in general terms before moving to the specific application to geographical variation.

3.1 Cascading Topic Models

Cascading topic models generate text from a chain of random variables. Each element in the chain defines a distribution over words, and acts as the mean of the distribution over the subsequent element in the chain. Thus, each element in the chain can be thought of as introducing some additional corruption. All words are drawn from the final distribution in the chain.

At the beginning of the chain are the priors, followed by unadulerated base topics, which may then be corrupted by other factors (such as geography or time). For example, consider a base "**food**" topic

[3] `http://dev.twitter.com/pages/streaming_api`

[4] `http://www.ark.cs.cmu.edu/GeoTwitter`

[5] `http://tweetmotif.com`

[6] The region could be observed by using a predefined geographical decomposition, e.g., political boundaries. However, such regions may not correspond well to linguistic variation.

that emphasizes words like *dinner* and *delicious*; the corrupted "**food-California**" topic would place weight on these words, but might place extra emphasis on other words like *sprouts*.

The path through the cascade is determined by a set of indexing variables, which may be hidden or observed. As in standard latent Dirichlet allocation (Blei et al., 2003), the base topics are selected by a per-token hidden variable z. In the geographical topic model, the next level corresponds to regions, which are selected by a per-author latent variable r.

Formally, we draw each level of the cascade from a normal distribution centered on the previous level; the final multinomial distribution over words is obtained by exponentiating and normalizing. To ensure tractable inference, we assume that all covariance matrices are uniform diagonal, i.e., $a\mathbf{I}$ with $a > 0$; this means we do not model interactions between words.

3.2 The Geographic Topic Model

The application of cascading topic models to geographical variation is straightforward. Each document corresponds to the entire Twitter feed of a given author during the time period covered by our corpus. For each author, the latent variable r corresponds to the geographical region of the author, which is not observed. As described above, r selects a corrupted version of each topic: the kth basic topic has mean $\boldsymbol{\mu}_k$, with uniform diagonal covariance σ_k^2; for region j, we can draw the regionally-corrupted topic from the normal distribution, $\boldsymbol{\eta}_{jk} \sim \mathcal{N}(\boldsymbol{\mu}_k, \sigma_k^2\mathbf{I})$.

Because $\boldsymbol{\eta}$ is normally-distributed, it lies not in the simplex but in $\mathbb{R}^W$. We deterministically compute multinomial parameters $\boldsymbol{\beta}$ by exponentiating and normalizing: $\boldsymbol{\beta}_{jk} = \exp(\boldsymbol{\eta}_{jk}) / \sum_i \exp(\eta_{jk}^{(i)})$. This normalization could introduce identifiability problems, as there are multiple settings for $\boldsymbol{\eta}$ that maximize $P(\boldsymbol{w}|\boldsymbol{\eta})$ (Blei and Lafferty, 2006a). However, this difficulty is obviated by the priors: given $\boldsymbol{\mu}$ and σ^2, there is only a single $\boldsymbol{\eta}$ that maximizes $P(\boldsymbol{w}|\boldsymbol{\eta})P(\boldsymbol{\eta}|\boldsymbol{\mu}, \sigma^2)$; similarly, only a single $\boldsymbol{\mu}$ maximizes $P(\boldsymbol{\eta}|\boldsymbol{\mu})P(\boldsymbol{\mu}|\boldsymbol{a}, b^2)$.

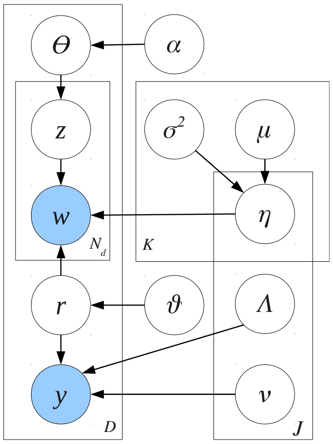
The observed latitude and longitude, denoted $\boldsymbol{y}$, are normally distributed and conditioned on the region, with mean $\boldsymbol{\nu}_r$ and precision matrix Λ_r indexed by the region r. The region index r is itself drawn from a single shared multinomial $\boldsymbol{\vartheta}$. The model is shown as a plate diagram in Figure 1.

Given a vocabulary size W, the generative story is as follows:

- **Generate base topics**: for each topic $k < K$
  - Draw the base topic from a normal distribution with uniform diagonal covariance: $\boldsymbol{\mu}_k \sim \mathcal{N}(\boldsymbol{a}, b^2\mathbf{I})$,
  - Draw the regional variance from a Gamma distribution: $\sigma_k^2 \sim \mathcal{G}(c, d)$.
  - **Generate regional variants**: for each region $j < J$,
    - * Draw the region-topic $\boldsymbol{\eta}_{jk}$ from a normal distribution with uniform diagonal covariance: $\boldsymbol{\eta}_{jk} \sim \mathcal{N}(\boldsymbol{\mu}_k, \sigma_k^2\mathbf{I})$.
    - * Convert $\boldsymbol{\eta}_{jk}$ into a multinomial distribution over words by exponentiating and normalizing: $\boldsymbol{\beta}_{jk} = \exp\left(\boldsymbol{\eta}_{jk}\right) / \sum_i^W \exp(\eta_{jk}^{(i)})$, where the denominator sums over the vocabulary.
- **Generate regions**: for each region $j < J$,
  - Draw the spatial mean $\boldsymbol{\nu}_j$ from a normal distribution.
  - Draw the precision matrix Λ_j from a Wishart distribution.
- Draw the distribution over regions $\boldsymbol{\vartheta}$ from a symmetric Dirichlet prior, $\boldsymbol{\vartheta} \sim \text{Dir}(\alpha_{\vartheta}\mathbf{1})$.
- **Generate text and locations**: for each document d,
  - Draw topic proportions from a symmetric Dirichlet prior, $\boldsymbol{\theta} \sim \text{Dir}(\alpha\mathbf{1})$.
  - Draw the region r from the multinomial distribution $\boldsymbol{\vartheta}$.
  - Draw the location $\boldsymbol{y}$ from the bivariate Gaussian, $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\nu}_r, \Lambda_r)$.
  - For each word token,
    - * Draw the topic indicator $z \sim \boldsymbol{\theta}$.
    - * Draw the word token $w \sim \boldsymbol{\beta}_{rz}$.

4 Inference

We apply mean-field variational inference: a fully-factored variational distribution Q is chosen to minimize the Kullback-Leibler divergence from the true distribution. Mean-field variational inference with conjugate priors is described in detail elsewhere (Bishop, 2006; Wainwright and Jordan, 2008); we restrict our focus to the issues that are unique to the geographic topic model.



$\boldsymbol{\mu}_k$	log of base topic k's distribution over word types
σ_k^2	variance parameter for regional variants of topic k
$\boldsymbol{\eta}_{jk}$	region j's variant of base topic $\boldsymbol{\mu}_k$
$\boldsymbol{\theta}_d$	author d's topic proportions
r_d	author d's latent region
$\boldsymbol{y}_d$	author d's observed GPS location
$\boldsymbol{\nu}_j$	region j's spatial center
Λ_j	region j's spatial precision
z_n	token n's topic assignment
w_n	token n's observed word type
$\boldsymbol{\alpha}$	global prior over author-topic proportions
$\boldsymbol{\vartheta}$	global prior over region classes

Figure 1: Plate diagram for the geographic topic model, with a table of all random variables. Priors (besides α) are omitted for clarity, and the document indices on z and w are implicit.

We place variational distributions over all latent variables of interest: $\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{r}, \boldsymbol{\vartheta}, \boldsymbol{\eta}, \boldsymbol{\mu}, \sigma^2, \boldsymbol{\nu}$, and Λ, updating each of these distributions in turn, until convergence. The variational distributions over $\boldsymbol{\theta}$ and $\boldsymbol{\vartheta}$ are Dirichlet, and have closed form updates: each can be set to the sum of the expected counts, plus a term from the prior (Blei et al., 2003). The variational distributions $q(z)$ and $q(r)$ are categorical, and can be set proportional to the expected joint likelihood—to set $q(z)$ we marginalize over r, and vice versa.[7] The updates for the multivariate Gaussian spatial parameters $\boldsymbol{\nu}$ and Λ are described by Penny (2001).

4.1 Regional Word Distributions

The variational region-topic distribution $\boldsymbol{\eta}_{jk}$ is normal, with uniform diagonal covariance for tractability. Throughout we will write $\langle x \rangle$ to indicate the expectation of x under the variational distribution Q. Thus, the vector mean of the distribution $q(\boldsymbol{\eta}_{jk})$ is written $\langle \boldsymbol{\eta}_{jk} \rangle$, while the variance (uniform across i) of $q(\boldsymbol{\eta})$ is written $\mathcal{V}(\boldsymbol{\eta}_{jk})$.

To update the mean parameter $\langle \boldsymbol{\eta}_{jk} \rangle$, we maximize the contribution to the variational bound L from the relevant terms:

$$L_{[\langle \eta_{jk}^{(i)} \rangle]} = \langle \log p(\boldsymbol{w}|\boldsymbol{\beta}, \boldsymbol{z}, \boldsymbol{r}) \rangle + \langle \log p(\eta_{jk}^{(i)} | \mu_k^{(i)}, \sigma_k^2) \rangle, \tag{1}$$

with the first term representing the likelihood of the observed words (recall that β is computed deterministically from η) and the second term corresponding to the prior. The likelihood term requires the expectation $\langle \log \boldsymbol{\beta} \rangle$, but this is somewhat complicated by the normalizer $\sum_i^W \exp(\eta^{(i)})$, which sums over all terms in the vocabulary. As in previous work on logistic normal topic models, we use a Taylor approximation for this term (Blei and Lafferty, 2006a).

The prior on $\boldsymbol{\eta}$ is normal, so the contribution from the second term of the objective (Equation 1) is $-\frac{1}{2\langle \sigma_k^2 \rangle} \langle (\eta_{jk}^{(i)} - \mu_k^{(i)})^2 \rangle$. We introduce the following notation for expected counts: $N(i, j, k)$ indicates the expected count of term i in region j and topic k, and $N(j, k) = \sum_i N(i, j, k)$. After some calculus, we can write the gradient $\partial L / \partial \langle \eta_{jk}^{((i))} \rangle$ as

$$N(i, j, k) - N(j, k) \langle \beta_{jk}^{(i)} \rangle - \langle \sigma_k^{-2} \rangle (\langle \eta_{jk}^{(i)} \rangle - \langle \mu_k^{(i)} \rangle), \tag{2}$$

which has an intuitive interpretation. The first two terms represent the difference in expected counts for term i under the variational distributions $q(z, r)$ and $q(z, r, \beta)$: this difference goes to zero when $\beta_{jk}^{(i)}$ perfectly matches $N(i, j, k)/N(j, k)$. The third term penalizes $\eta_{jk}^{(i)}$ for deviating from its prior $\mu_k^{(i)}$, but this penalty is proportional to the expected inverse variance $\langle \sigma_k^{-2} \rangle$. We apply gradient ascent to maximize the objective L. A similar set of calculations gives the gradient for the variance of $\boldsymbol{\eta}$; these are described in an forthcoming appendix.

[7] Thanks to the naïve mean field assumption, we can marginalize over $\boldsymbol{z}$ by first decomposing across all N_d words and then summing over $q(z)$.

4.2 Base Topics

The base topic parameters are $\boldsymbol{\mu}_k$ and σ_k^2; in the variational distribution, $q(\boldsymbol{\mu}_k)$ is normally distributed and $q(\sigma_k^2)$ is Gamma distributed. Note that $\boldsymbol{\mu}_k$ and σ_k^2 affect only the regional word distributions $\boldsymbol{\eta}_{jk}$. An advantage of the logistic normal is that the variational parameters over $\boldsymbol{\mu}_k$ are available in closed form,

$$\langle \mu_k^{(i)} \rangle = \frac{b^2 \sum_j^J \langle \eta_{jk}^{(i)} \rangle + \langle \sigma_k^2 \rangle a^{(i)}}{b^2 J + \langle \sigma_k^2 \rangle}$$
$$\mathcal{V}(\boldsymbol{\mu}_k) = (b^{-2} + J \langle \sigma_k^{-2} \rangle)^{-1},$$

where J indicates the number of regions. The expectation of the base topic μ incorporates the prior and the average of the generated region-topics—these two components are weighted respectively by the expected variance of the region-topics $\langle \sigma_k^2 \rangle$ and the prior topical variance b^2. The posterior variance $\mathcal{V}(\boldsymbol{\mu})$ is a harmonic combination of the prior variance b^2 and the expected variance of the region topics.

The variational distribution over the region-topic variance σ_k^2 has Gamma parameters. These parameters cannot be updated in closed form, so gradient optimization is again required. The derivation of these updates is more involved, and is left for a forthcoming appendix.

5 Implementation

Variational scheduling and initialization are important aspects of any hierarchical generative model, and are often under-discussed. In our implementation, the variational updates are scheduled as follows: given expected counts, we iteratively update the variational parameters on the region-topics $\boldsymbol{\eta}$ and the base topics $\boldsymbol{\mu}$, until convergence. We then update the geographical parameters $\boldsymbol{\nu}$ and Λ, as well as the distribution over regions $\boldsymbol{\vartheta}$. Finally, for each document we iteratively update the variational parameters over $\boldsymbol{\theta}, \boldsymbol{z}$, and r until convergence, obtaining expected counts that are used in the next iteration of updates for the topics and their regional variants. We iterate an outer loop over the entire set of updates until convergence.

We initialize the model in a piecewise fashion. First we train a Dirichlet process mixture model on the locations $\boldsymbol{y}$, using variational inference on the truncated stick-breaking approximation (Blei and Jordan, 2006). This automatically selects the number of regions J, and gives a distribution over each region indicator r_d from geographical information alone. We then run standard latent Dirichlet allocation to obtain estimates of $\boldsymbol{z}$ for each token (ignoring the locations). From this initialization we can compute the first set of expected counts, which are used to obtain initial estimates of all parameters needed to begin variational inference in the full model.

The prior $\boldsymbol{a}$ is the expected mean of each topic $\boldsymbol{\mu}$; for each term i, we set $a^{(i)} = \log N(i) - \log N$, where $N(i)$ is the total count of i in the corpus and $N = \sum_i N(i)$. The variance prior b^2 is set to 1, and the prior on σ^2 is the Gamma distribution $\mathcal{G}(2, 200)$, encouraging minimal deviation from the base topics. The symmetric Dirichlet prior on $\boldsymbol{\theta}$ is set to $\frac{1}{2}$, and the symmetric Dirichlet parameter on ϑ is updated from weak hyperpriors (Minka, 2003). Finally, the geographical model takes priors that are linked to the data: for each region, the mean is very weakly encouraged to be near the overall mean, and the covariance prior is set by the average covariance of clusters obtained by running K-means.

6 Evaluation

For a quantitative evaluation of the estimated relationship between text and geography, we assess our model's ability to predict the geographic location of unlabeled authors based on their text alone.[8] This task may also be practically relevant as a step toward applications for recommending local businesses or social connections. A randomly-chosen 60% of authors are used for training, 20% for development, and the remaining 20% for final evaluation.

6.1 Systems

We compare several approaches for predicting author location; we divide these into latent variable generative models and discriminative approaches.

[8] Alternatively, one might evaluate the attributed regional memberships of the words themselves. While the Dictionary of American Regional English (Cassidy and Hall, 1985) attempts a comprehensive list of all regionally-affiliated terms, it is based on interviews conducted from 1965-1970, and the final volume (covering Si–Z) is not yet complete.

6.1.1 Latent Variable Models

Geographic Topic Model This is the full version of our system, as described in this paper. To predict the unseen location $\boldsymbol{y}_d$, we iterate until convergence on the variational updates for the hidden topics $\boldsymbol{z}_d$, the topic proportions $\boldsymbol{\theta}_d$, and the region r_d. From r_d, the location can be estimated as $\hat{\boldsymbol{y}}_d = \arg\max_{\boldsymbol{y}} \sum_j^J p(\boldsymbol{y}|\boldsymbol{\nu}_j, \Lambda_j) q(r_d = j)$. The development set is used to tune the number of topics and to select the best of multiple random initializations.

Mixture of Unigrams A core premise of our approach is that modeling topical variation will improve our ability to understand geographical variation. We test this idea by fixing $K = 1$, running our system with only a single topic. This is equivalent to a Bayesian mixture of unigrams in which each author is assigned a single, regional unigram language model that generates all of his or her text. The development set is used to select the best of multiple random initializations.

Supervised Latent Dirichlet Allocation In a more subtle version of the mixture-of-unigrams model, we model each author as an admixture of regions. Thus, the latent variable attached to each author is no longer an index, but rather a vector on the simplex. This model is equivalent to supervised latent Dirichlet allocation (Blei and McAuliffe, 2007): each topic is associated with equivariant Gaussian distributions over the latitude and longitude, and these topics must explain both the text and the observed geographical locations. For unlabeled authors, we estimate latitude and longitude by estimating the topic proportions and then applying the learned geographical distributions. This is a linear prediction

$$f(\bar{\boldsymbol{z}}_d; \boldsymbol{a}) = (\bar{\boldsymbol{z}}_d^{\mathsf{T}} \boldsymbol{a}^{\text{lat}},\ \bar{\boldsymbol{z}}_d^{\mathsf{T}} \boldsymbol{a}^{\text{lon}})$$

for an author's topic proportions $\bar{\boldsymbol{z}}_d$ and topic-geography weights $\boldsymbol{a} \in \mathbb{R}^{2K}$.

6.1.2 Baseline Approaches

Text Regression We perform linear regression to discriminatively learn the relationship between words and locations. Using term frequency features $\boldsymbol{x}_d$ for each author, we predict locations with word-geography weights $\boldsymbol{a} \in \mathbb{R}^{2W}$:

$$f(\boldsymbol{x}_d; \boldsymbol{a}) = (\boldsymbol{x}_d^{\mathsf{T}} \boldsymbol{a}^{\text{lat}},\ \boldsymbol{x}_d^{\mathsf{T}} \boldsymbol{a}^{\text{lon}})$$

Weights are trained to minimize the sum of squared Euclidean distances, subject to L_1 regularization:

$$\sum_d (\boldsymbol{x}_d^{\mathsf{T}} \boldsymbol{a}^{\text{lat}} - y_d^{\text{lat}})^2 + (\boldsymbol{x}_d^{\mathsf{T}} \boldsymbol{a}^{\text{lon}} - y_d^{\text{lon}})^2$$
$$+ \lambda_{\text{lat}} ||\boldsymbol{a}^{\text{lat}}||_1 + \lambda_{\text{lon}} ||\boldsymbol{a}^{\text{lon}}||_1$$

The minimization problem decouples into two separate latitude and longitude models, which we fit using the `glmnet` elastic net regularized regression package (Friedman et al., 2010), which obtained good results on other text-based prediction tasks (Joshi et al., 2010). Regularization parameters were tuned on the development set. The L_1 penalty outperformed L_2 and mixtures of L_1 and L_2.

Note that for both word-level linear regression here, and the topic-level linear regression in SLDA, the choice of squared Euclidean distance dovetails with our use of spatial Gaussian likelihoods in the geographic topic models, since optimizing $\boldsymbol{a}$ is equivalent to maximum likelihood estimation under the assumption that locations are drawn from equivariant circular Gaussians centered around each $f(\boldsymbol{x}_d; \boldsymbol{a})$ linear prediction. We experimented with decorrelating the location dimensions by projecting $\boldsymbol{y}_d$ into the principal component space, but this did not help text regression.

***K*-Nearest Neighbors** Linear regression is a poor model for the multimodal density of human populations. As an alternative baseline, we applied supervised K-nearest neighbors to predict the location $\boldsymbol{y}_d$ as the average of the positions of the K most similar authors in the training set. We computed term-frequency inverse-document frequency features and applied cosine similarity over their first 30 principal components to find the neighbors. The choices of principal components, IDF weighting, and neighborhood size $K = 20$ were tuned on the development set.

6.2 Metrics

Our principle error metrics are the mean and median distance between the predicted and true location in kilometers.[9] Because the distance error may be difficult to interpret, we also report accuracy of classi-

[9] For convenience, model training and prediction use latitude and longitude as an unprojected 2D Euclidean space. However, properly measuring the physical distance between points on the

System	Regression		Classification accuracy (%)	
	Mean Dist. (km)	Median Dist. (km)	Region (4-way)	State (49-way)
Geographic topic model	**900**	**494**	**58**	24
Mixture of unigrams	947	644	53	19
Supervised LDA	1055	728	39	4
Text regression	948	712	41	4
K-nearest neighbors	1077	853	37	2
Mean location	1148	1018		
Most common class			37	**27**

Table 1: Location prediction results; lower scores are better on the regression task, higher scores are better on the classification task. Distances are in kilometers. Mean location and most common class are computed from the test set. Both the geographic topic model and supervised LDA use the best number of topics from the development set (10 and 5, respectively).

fication by state and by region of the United States. Our data includes the 48 contiguous states plus the District of Columbia; the U.S. Census Bureau divides these states into four regions: West, Midwest, Northeast, and South.[10] Note that while major population centers straddle several state lines, most region boundaries are far from the largest cities, resulting in a clearer analysis.

6.3 Results

As shown in Table 1, the geographic topic model achieves the strongest performance on all metrics. All differences in performance between systems are statistically significant ($p < .01$) using the Wilcoxon-Mann-Whitney test for regression error and the χ^2 test for classification accuracy. Figure 2 shows how performance changes as the number of topics varies.

Note that the geographic topic model and the mixture of unigrams use identical code and parametrization – the only difference is that the geographic topic model accounts for topical variation, while the mixture of unigrams sets $K = 1$. These results validate our basic premise that it is important to model the interaction between topical and geographical variation.

Text regression and supervised LDA perform especially poorly on the classification metric. Both methods make predictions that are averaged across each word in the document: in text regression, each word is directly multiplied by a feature weight; in supervised LDA the word is associated with a latent topic first, and then multiplied by a weight. For these models, all words exert an influence on the predicted location, so uninformative words will draw the prediction towards the center of the map. This yields reasonable distance errors but poor classification accuracy. We had hoped that K-nearest neighbors would be a better fit for this metric, but its performance is poor at all values of K. Of course it is always possible to optimize classification accuracy directly, but such an approach would be incapable of predicting the exact geographical location, which is the focus of our evaluation (given that the desired geographical partition is unknown). Note that the geographic topic model is also not trained to optimize classification accuracy.

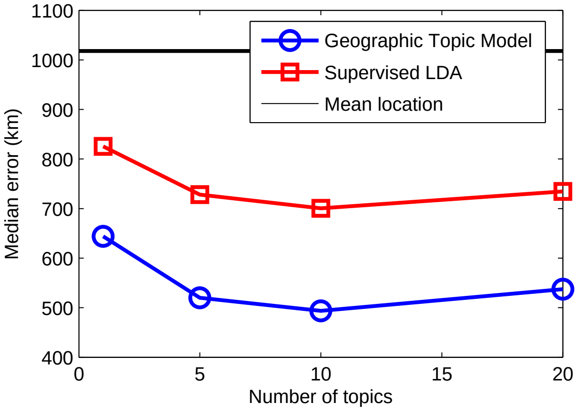


Figure 2: The effect of varying the number of topics on the median regression error (lower is better).

Earth's surface requires computing or approximating the great circle distance – we use the Haversine formula (Sinnott, 1984). For the continental U.S., the relationship between degrees and kilometers is nearly linear, but extending the model to a continental scale would require a more sophisticated approach.

[10] http://www.census.gov/geo/www/us_regdiv.pdf

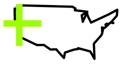
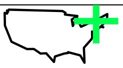
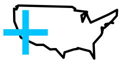
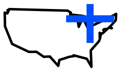
	“basketball”	**“popular music”**	**“daily life”**	**“emoticons”**	**“chit chat”**
	PISTONS KOBE LAKERS game DUKE NBA CAVS STUCKEY JETS KNICKS	album music beats artist video #LAKERS ITUNES tour produced vol	tonight shop weekend getting going chilling ready discount waiting iam	:) haha :d :(;) :p xd :/ hahaha hahah	lol smh jk yea wyd coo ima wassup somethin jp
Boston	CELTICS victory BOSTON CHARLOTTE	playing daughter PEARL alive war comp	BOSTON	;p gna loveee	*ese* exam suttin sippin
N. California 	THUNDER KINGS GIANTS pimp trees clap	SIMON dl mountain seee	6am OAKLAND	*pues* hella koo SAN fckn	hella flirt hut iono OAKLAND
New York 	NETS KNICKS	BRONX	iam cab	oww	wasssup nm
Los Angeles 	#KOBE #LAKERS AUSTIN	#LAKERS load HOLLYWOOD imm MICKEY TUPAC	omw tacos hr HOLLYWOOD	af *papi* raining th bomb coo HOLLYWOOD	wyd coo af *nada* tacos messin fasho bomb
Lake Erie 	CAVS CLEVELAND OHIO BUCKS od COLUMBUS	premiere prod joint TORONTO onto designer CANADA village burr	stink CHIPOTLE tipsy	;d blvd BIEBER hve OHIO	foul WIZ salty excuses lames officer lastnight

Table 2: Example base topics (top line) and regional variants. For the base topics, terms are ranked by log-odds compared to the background distribution. The regional variants show words that are strong compared to both the base topic and the background. Foreign-language words are shown in *italics*, while terms that are usually in proper nouns are shown in SMALL CAPS. See Table 3 for definitions of slang terms; see Section 7 for more explanation and details on the methodology.

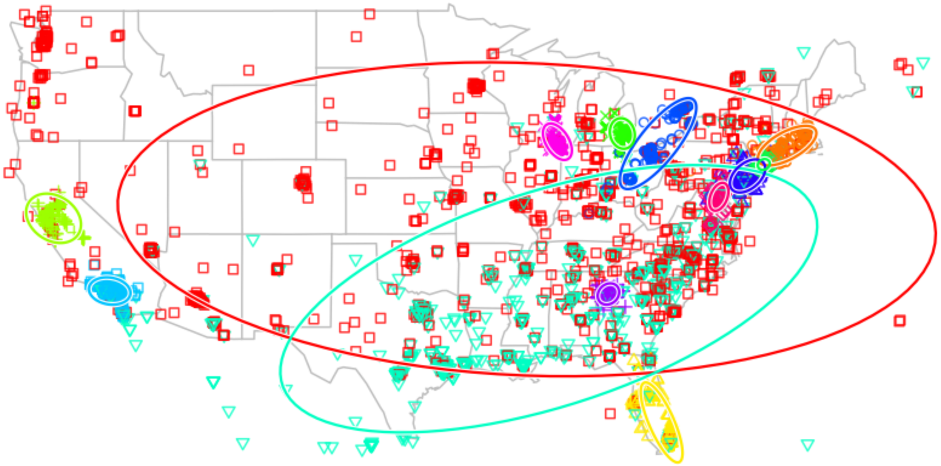


Figure 3: Regional clustering of the training set obtained by one randomly-initialized run of the geographical topic model. Each point represents one author, and each shape/color combination represents the most likely cluster assignment. Ellipses represent the regions' spatial means and covariances. The same model and coloring are shown in Table 2.

7 Analysis

Our model permits analysis of geographical variation in the context of topics that help to clarify the significance of geographically-salient terms. Table 2 shows a subset of the results of one randomly-initialized run, including five hand-chosen topics (of 50 total) and five regions (of 13, as chosen automatically during initialization). Terms were selected by log-odds comparison. For the base topics we show the ten strongest terms in each topic as compared to the background word distribution. For the regional variants, we show terms that are strong both regionally and topically: specifically, we select terms that are in the top 100 compared to both the background distribution and to the base topic. The names for the topics and regions were chosen by the authors.

Nearly all of the terms in column 1 ("**basketball**") refer to sports teams, athletes, and place names—encouragingly, terms tend to appear in the regions where their referents reside. Column 2 contains several proper nouns, mostly referring to popular music figures (including PEARL from the band Pearl Jam).[11] Columns 3–5 are more conversational. Spanish-language terms (*papi, pues, nada, ese*) tend to appear in regions with large Spanish-speaking populations—it is also telling that these terms appear in topics with emoticons and slang abbreviations, which may transcend linguistic barriers. Other terms refer to people or subjects that may be especially relevant in certain regions: *tacos* appears in the southern California region and *cab* in the New York region; TUPAC refers to a rap musician from Los Angeles, and WIZ refers to a rap musician from Pittsburgh, not far from the center of the "Lake Erie" region.

A large number of slang terms are found to have strong regional biases, suggesting that slang may depend on geography more than standard English does. The terms *af* and *hella* display especially strong regional affinities, appearing in the regional variants of multiple topics (see Table 3 for definitions). Northern and Southern California use variant spellings *koo* and *coo* to express the same meaning.

term	definition	term	definition
af	as fuck (very)	jk	just kidding
coo	cool	jp	just playing (kidding)
dl	download		
fasho	for sure	koo	cool
gna	going to	lol	laugh out loud
hella	very	nm	nothing much
hr	hour	od	overdone (very)
iam	I am	omw	on my way
ima	I'm going to	smh	shake my head
imm	I'm	suttin	something
iono	I don't know	wassup	what's up
lames	lame (not cool) people	wyd	what are you doing?

Table 3: A glossary of non-standard terms from Table 2. Definitions are obtained by manually inspecting the context in which the terms appear, and by consulting `www.urbandictionary.com`.

While research in perceptual dialectology does confirm the link of *hella* to Northern California (Bucholtz et al., 2007), we caution that our findings are merely suggestive, and a more rigorous analysis must be undertaken before making definitive statements about the regional membership of individual terms. We view the geographic topic model as an exploratory tool that may be used to facilitate such investigations.

Figure 3 shows the regional clustering on the training set obtained by one run of the model. Each point represents an author, and the ellipses represent the bivariate Gaussians for each region. There are nine compact regions for major metropolitan areas, two slightly larger regions that encompass Florida and the area around Lake Erie, and two large regions that partition the country roughly into north and south.

8 Related Work

The relationship between language and geography has been a topic of interest to linguists since the nineteenth century (Johnstone, 2010). An early work of particular relevance is Kurath's *Word Geography of the Eastern United States* (1949), in which he conducted interviews and then mapped the occurrence of equivalent word pairs such as *stoop* and *porch*. The essence of this approach—identifying variable pairs and measuring their frequencies—remains a dominant methodology in both dialec-

[11] This analysis is from an earlier version of our dataset that contained some Twitterbots, including one from a Boston-area radio station. The bots were purged for the evaluation in Section 6, though the numerical results are nearly identical.

tology (Labov et al., 2006) and sociolinguistics (Tagliamonte, 2006). Within this paradigm, computational techniques are often applied to post hoc analysis: logistic regression (Sankoff et al., 2005) and mixed-effects models (Johnson, 2009) are used to measure the contribution of individual variables, while hierarchical clustering and multidimensional scaling enable aggregated inference across multiple variables (Nerbonne, 2009). However, in all such work it is assumed that the relevant linguistic variables have already been identified—a time-consuming process involving considerable linguistic expertise. We view our work as complementary to this tradition: we work directly from raw text, identifying both the relevant features and coherent linguistic communities.

An active recent literature concerns geotagged information on the web, such as search queries (Backstrom et al., 2008) and tagged images (Crandall et al., 2009). This research identifies the geographic distribution of individual queries and tags, but does not attempt to induce any structural organization of either the text or geographical space, which is the focus of our research. More relevant is the work of Mei et al. (2006), in which the distribution over latent topics in blog posts is conditioned on the geographical location of the author. This is somewhat similar to the supervised LDA model that we consider, but their approach assumes that a partitioning of geographical space into regions is already given.

Methodologically, our cascading topic model is designed to capture multiple dimensions of variability: topics and geography. Mei et al. (2007) include sentiment as a second dimension in a topic model, using a switching variable so that individual word tokens may be selected from either the topic or the sentiment. However, our hypothesis is that individual word tokens reflect both the topic and the geographical aspect. Sharing this intuition, Paul and Girju (2010) build topic-aspect models for the cross product of topics and aspects. They do not impose any regularity across multiple aspects of the same topic, so this approach may not scale when the number of aspects is large (they consider only two aspects). We address this issue using cascading distributions; when the observed data for a given region-topic pair is low, the model falls back to the base topic. The use of cascading logistic normal distributions in topic models follows earlier work on dynamic topic models (Blei and Lafferty, 2006b; Xing, 2005).

9 Conclusion

This paper presents a model that jointly identifies words with high regional affinity, geographically-coherent linguistic regions, and the relationship between regional and topic variation. The key modeling assumption is that regions and topics interact to shape observed lexical frequencies. We validate this assumption on a prediction task in which our model outperforms strong alternatives that do not distinguish regional and topical variation.

We see this work as a first step towards a unsupervised methodology for modeling linguistic variation using raw text. Indeed, in a study of morphosyntactic variation, Szmrecsanyi (2010) finds that by the most generous measure, geographical factors account for only 33% of the observed variation. Our analysis might well improve if non-geographical factors were considered, including age, race, gender, income and whether a location is urban or rural. In some regions, estimates of many of these factors may be obtained by cross-referencing geography with demographic data. We hope to explore this possibility in future work.

Acknowledgments

We would like to thank Amr Ahmed, Jonathan Chang, Shay Cohen, William Cohen, Ross Curtis, Miro Dudík, Scott Kiesling, Seyoung Kim, and the anonymous reviewers. This research was enabled by Google's support of the Worldly Knowledge project at CMU, AFOSR FA9550010247, ONR N0001140910758, NSF CAREER DBI-0546594, NSF IIS-0713379, and an Alfred P. Sloan Fellowship.

References

L. Backstrom, J. Kleinberg, R. Kumar, and J. Novak. 2008. Spatial variation in search engine queries. In *Proceedings of WWW*.

C. M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

D. M. Blei and M. I. Jordan. 2006. Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 1:121–144.

D. M. Blei and J. Lafferty. 2006a. Correlated topic models. In *NIPS*.

D. M. Blei and J. Lafferty. 2006b. Dynamic topic models. In *Proceedings of ICML*.

D. M. Blei and J. D. McAuliffe. 2007. Supervised topic models. In *NIPS*.

D. M. Blei, A. Y. Ng, and M. I. Jordan. 2003. Latent Dirichlet allocation. *JMLR*, 3:993–1022.

M. Bucholtz, N. Bermudez, V. Fung, L. Edwards, and R. Vargas. 2007. Hella Nor Cal or totally So Cal? the perceptual dialectology of California. *Journal of English Linguistics*, 35(4):325–352.

F. G. Cassidy and J. H. Hall. 1985. *Dictionary of American Regional English*, volume 1. Harvard University Press.

J. Chambers. 2009. *Sociolinguistic Theory: Linguistic Variation and its Social Significance*. Blackwell.

D. J Crandall, L. Backstrom, D. Huttenlocher, and J. Kleinberg. 2009. Mapping the world's photos. In *Proceedings of WWW*, page 761770.

J. Friedman, T. Hastie, and R. Tibshirani. 2010. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1).

D. E. Johnson. 2009. Getting off the GoldVarb standard: Introducing Rbrul for mixed-effects variable rule analysis. *Language and Linguistics Compass*, 3(1):359–383.

B. Johnstone. 2010. Language and place. In R. Mesthrie and W. Wolfram, editors, *Cambridge Handbook of Sociolinguistics*. Cambridge University Press.

M. Joshi, D. Das, K. Gimpel, and N. A. Smith. 2010. Movie reviews and revenues: An experiment in text regression. In *Proceedings of NAACL-HLT*.

H. Kurath. 1949. *A Word Geography of the Eastern United States*. University of Michigan Press.

H. Kwak, C. Lee, H. Park, and S. Moon. 2010. What is Twitter, a social network or a news media? In *Proceedings of WWW*.

W. Labov, S. Ash, and C. Boberg. 2006. *The Atlas of North American English: Phonetics, Phonology, and Sound Change*. Walter de Gruyter.

W. Labov. 1966. *The Social Stratification of English in New York City*. Center for Applied Linguistics.

Q. Mei, C. Liu, H. Su, and C. X Zhai. 2006. A probabilistic approach to spatiotemporal theme pattern mining on weblogs. In *Proceedings of WWW*, page 542.

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 WWW*.

T. P. Minka. 2003. Estimating a Dirichlet distribution. Technical report, Massachusetts Institute of Technology.

J. Nerbonne. 2009. Data-driven dialectology. *Language and Linguistics Compass*, 3(1).

B. O'Connor, M. Krieger, and D. Ahn. 2010. TweetMotif: Exploratory search and topic summarization for twitter. In *Proceedings of ICWSM*.

J. C. Paolillo. 2002. *Analyzing Linguistic Variation: Statistical Models and Methods*. CSLI Publications.

M. Paul and R. Girju. 2010. A two-dimensional topic-aspect model for discovering multi-faceted topics. In *Proceedings of AAAI*.

W. D. Penny. 2001. Variational Bayes for d-dimensional Gaussian mixture models. Technical report, University College London.

D. Sankoff, S. A. Tagliamonte, and E. Smith. 2005. Goldvarb X: A variable rule application for Macintosh and Windows. Technical report, Department of Linguistics, University of Toronto.

R. W. Sinnott. 1984. Virtues of the Haversine. *Sky and Telescope*, 68(2).

B. Szmrecsanyi. 2010. Geography is overrated. In S. Hansen, C. Schwarz, P. Stoeckle, and T. Streck, editors, *Dialectological and Folk Dialectological Concepts of Space*. Walter de Gruyter.

S. A. Tagliamonte and D. Denis. 2008. Linguistic ruin? LOL! Instant messanging and teen language. *American Speech*, 83.

S. A. Tagliamonte. 2006. *Analysing Sociolinguistic Variation*. Cambridge University Press.

M. J. Wainwright and M. I. Jordan. 2008. *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers.

E. P. Xing. 2005. On topic evolution. Technical Report 05-115, Center for Automated Learning and Discovery, Carnegie Mellon University.

Dual Decomposition for Parsing with Non-Projective Head Automata

Terry Koo Alexander M. Rush Michael Collins Tommi Jaakkola David Sontag
MIT CSAIL, Cambridge, MA 02139, USA
{maestro,srush,mcollins,tommi,dsontag}@csail.mit.edu

Abstract

This paper introduces algorithms for non-projective parsing based on *dual decomposition*. We focus on parsing algorithms for *non-projective head automata*, a generalization of head-automata models to non-projective structures. The dual decomposition algorithms are simple and efficient, relying on standard dynamic programming and minimum spanning tree algorithms. They provably solve an LP relaxation of the non-projective parsing problem. Empirically the LP relaxation is very often tight: for many languages, exact solutions are achieved on over 98% of test sentences. The accuracy of our models is higher than previous work on a broad range of datasets.

1 Introduction

Non-projective dependency parsing is useful for many languages that exhibit non-projective syntactic structures. Unfortunately, the non-projective parsing problem is known to be NP-hard for all but the simplest models (McDonald and Satta, 2007). There has been a long history in combinatorial optimization of methods that exploit structure in complex problems, using methods such as *dual decomposition* or *Lagrangian relaxation* (Lemaréchal, 2001). Thus far, however, these methods are not widely used in NLP.

This paper introduces algorithms for non-projective parsing based on dual decomposition. We focus on parsing algorithms for *non-projective head automata*, a generalization of the head-automata models of Eisner (2000) and Alshawi (1996) to non-projective structures. These models include non-projective dependency parsing models with higher-order (e.g., sibling and/or grandparent) dependency relations as a special case. Although decoding of full parse structures with non-projective head automata is intractable, we leverage the observation that key components of the decoding *can* be efficiently computed using combinatorial algorithms. In particular,

1. Decoding for individual head-words can be accomplished using dynamic programming.
2. Decoding for arc-factored models can be accomplished using directed minimum-weight spanning tree (MST) algorithms.

The resulting parsing algorithms have the following properties:

- They are efficient and easy to implement, relying on standard dynamic programming and MST algorithms.
- They provably solve a linear programming (LP) relaxation of the original decoding problem.
- Empirically the algorithms very often give an exact solution to the decoding problem, in which case they also provide a certificate of optimality.

In this paper we first give the definition for non-projective head automata, and describe the parsing algorithm. The algorithm can be viewed as an instance of Lagrangian relaxation; we describe this connection, and give convergence guarantees for the method. We describe a generalization to models that include grandparent dependencies. We then introduce a perceptron-driven training algorithm that makes use of point 1 above.

We describe experiments on non-projective parsing for a number of languages, and in particular compare the dual decomposition algorithm to approaches based on general-purpose linear programming (LP) or integer linear programming (ILP) solvers (Martins et al., 2009). The accuracy of our models is higher than previous work on a broad range of datasets. The method gives exact solutions to the decoding problem, together with a certificate of optimality, on over 98% of test examples for many of the test languages, with parsing times ranging between 0.021 seconds/sentence for the most simple languages/models, to 0.295 seconds/sentence for the

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1288–1298,
MIT, Massachusetts, USA, 9-11 October 2010.

most complex settings. The method compares favorably to previous work using LP/ILP formulations, both in terms of efficiency, and also in terms of the percentage of exact solutions returned.

While the focus of the current paper is on non-projective dependency parsing, the approach opens up new ways of thinking about parsing algorithms for lexicalized formalisms such as TAG (Joshi and Schabes, 1997), CCG (Steedman, 2000), and projective head automata.

2 Related Work

McDonald et al. (2005) describe MST-based parsing for non-projective dependency parsing models with arc-factored decompositions; McDonald and Pereira (2006) make use of an approximate (hill-climbing) algorithm for parsing with more complex models. McDonald and Pereira (2006) and McDonald and Satta (2007) describe complexity results for non-projective parsing, showing that parsing for a variety of models is NP-hard. Riedel and Clarke (2006) describe ILP methods for the problem; Martins et al. (2009) recently introduced alternative LP and ILP formulations. Our algorithm differs in that we do not use general-purpose LP or ILP solvers, instead using an MST solver in combination with dynamic programming; thus we leverage the underlying structure of the problem, thereby deriving more efficient decoding algorithms.

Both dual decomposition and Lagrangian relaxation have a long history in combinatorial optimization. Our work was originally inspired by recent work on dual decomposition for inference in graphical models (Wainwright et al., 2005; Komodakis et al., 2007). However, the non-projective parsing problem has a very different structure from these models, and the decomposition we use is very different in nature from those used in graphical models. Other work has made extensive use of decomposition approaches for efficiently solving LP relaxations for graphical models (e.g., Sontag et al. (2008)). Methods that incorporate combinatorial solvers within loopy belief propagation (LBP) (Duchi et al., 2007; Smith and Eisner, 2008) are also closely related to our approach. Unlike LBP, our method has strong theoretical guarantees, such as guaranteed convergence and the possibility of a certificate of optimality.

Finally, in other recent work, Rush et al. (2010) describe dual decomposition approaches for other NLP problems.

3 Sibling Models

This section describes a particular class of models, *sibling models*; the next section describes a dual-decomposition algorithm for decoding these models.

Consider the dependency parsing problem for a sentence with n words. We define the *index set* for dependency parsing to be $\mathcal{I} = \{(i,j) : i \in \{0 \dots n\}, j \in \{1 \dots n\}, i \neq j\}$. A dependency parse is a vector $y = \{y(i,j) : (i,j) \in \mathcal{I}\}$, where $y(i,j) = 1$ if a dependency with head word i and modifier j is in the parse, 0 otherwise. We use $i = 0$ for the root symbol. We define $\mathcal{Y}$ to be the set of all well-formed non-projective dependency parses (i.e., the set of directed spanning trees rooted at node 0). Given a function $f : \mathcal{Y} \mapsto \mathbb{R}$ that assigns scores to parse trees, the optimal parse is

$$y^* = \operatorname*{argmax}_{y \in \mathcal{Y}} f(y) \qquad (1)$$

A particularly simple definition of $f(y)$ is $f(y) = \sum_{(i,j) \in \mathcal{I}} y(i,j)\theta(i,j)$ where $\theta(i,j)$ is the score for dependency (i,j). Models with this form are often referred to as *arc-factored models*. In this case the optimal parse tree y^* can be found efficiently using MST algorithms (McDonald et al., 2005).

This paper describes algorithms that compute y^* for more complex definitions of $f(y)$; in this section, we focus on algorithms for models that capture interactions between sibling dependencies. To this end, we will find it convenient to define the following notation. Given a vector y, define

$$y_{|i} = \{y(i,j) : j = 1 \dots n, j \neq i\}$$

Hence $y_{|i}$ specifies the set of modifiers to word i; note that the vectors $y_{|i}$ for $i = 0 \dots n$ form a partition of the full set of variables.

We then assume that $f(y)$ takes the form

$$f(y) = \sum_{i=0}^{n} f_i(y_{|i}) \qquad (2)$$

Thus $f(y)$ decomposes into a sum of terms, where each f_i considers modifiers to the i'th word alone.

In the general case, finding $y^* = \operatorname{argmax}_{y \in \mathcal{Y}} f(y)$ under this definition of $f(y)$ is an NP-hard problem. However for certain definitions of f_i, it is possible to efficiently compute $\operatorname{argmax}_{y_{|i} \in \mathcal{Z}_i} f_i(y_{|i})$ for any value of i, typically using dynamic programming. (Here we use $\mathcal{Z}_i$ to refer to the set of all possible values for $y_{|i}$: specifically, $\mathcal{Z}_0 = \{0,1\}^n$ and for $i \neq 0$, $\mathcal{Z}_i = \{0,1\}^{n-1}$.) In these cases we can efficiently compute

$$z^* = \operatorname*{argmax}_{z \in \mathcal{Z}} f(z) = \operatorname*{argmax}_{z \in \mathcal{Z}} \sum_i f_i(z_{|i}) \qquad (3)$$

where $\mathcal{Z} = \{z : z_{|i} \in \mathcal{Z}_i \quad \text{for } i = 0 \ldots n\}$ by simply computing $z^*_{|i} = \operatorname{argmax}_{z_{|i} \in \mathcal{Z}_i} f_i(z_{|i})$ for $i = 0 \ldots n$. Eq. 3 can be considered to be an approximation to Eq. 1, where we have replaced $\mathcal{Y}$ with $\mathcal{Z}$. We will make direct use of this approximation in the dual decomposition parsing algorithm. Note that $\mathcal{Y} \subseteq \mathcal{Z}$, and in all but trivial cases, $\mathcal{Y}$ is a strict subset of $\mathcal{Z}$. For example, a structure $z \in \mathcal{Z}$ could have $z(i,j) = z(j,i) = 1$ for some (i,j); it could contain longer cycles; or it could contain words that do not modify exactly one head. Nevertheless, with suitably powerful functions f_i—for example functions based on discriminative models—z^* may be a good approximation to y^*. Later we will see that dual decomposition can effectively use MST inference to rule out ill-formed structures.

We now give the main assumption underlying sibling models:

Assumption 1 (Sibling Decompositions) *A model $f(y)$ satisfies the sibling-decomposition assumption if: 1) $f(y) = \sum_{i=0}^{n} f_i(y_{|i})$ for some set of functions $f_0 \ldots f_n$. 2) For any $i \in \{0 \ldots n\}$, for any value of the variables $u(i,j) \in \mathbb{R}$ for $j = 1 \ldots n$, it is possible to compute*

$$\operatorname*{argmax}_{y_{|i} \in \mathcal{Z}_i} \left(f_i(y_{|i}) - \sum_j u(i,j) y(i,j) \right)$$

in polynomial time.

The second condition includes additional terms involving $u(i,j)$ variables that modify the scores of individual dependencies. These terms are benign for most definitions of f_i, in that they do not alter decoding complexity. They will be of direct use in the dual decomposition parsing algorithm.

Example 1: Bigram Sibling Models. Recall that $y_{|i}$ is a binary vector specifying which words are modifiers to the head-word i. Define $l_1 \ldots l_p$ to be the sequence of left modifiers to word i under $y_{|i}$, and $r_1 \ldots r_q$ to be the set of right modifiers (e.g., consider the case where $n = 5$, $i = 3$, and we have $y(3,1) = y(3,5) = 0$, and $y(3,2) = y(3,4) = 1$: in this case $p = 1$, $l_1 = 2$, and $q = 1$, $r_1 = 4$). In *bigram sibling models*, we have

$$f_i(y_{|i}) = \sum_{k=1}^{p+1} g_L(i, l_{k-1}, l_k) + \sum_{k=1}^{q+1} g_R(i, r_{k-1}, r_k)$$

where $l_0 = r_0 = \text{START}$ is the initial state, and $l_{p+1} = r_{q+1} = \text{END}$ is the end state. The functions g_L and g_R assign scores to bigram dependencies to the left and right of the head. Under this model calculating $\operatorname{argmax}_{y_{|i} \in \mathcal{Z}_i} \left(f_i(y_{|i}) - \sum_j u(i,j) y(i,j) \right)$ takes $O(n^2)$ time using dynamic programming, hence the model satisfies Assumption 1. □

Example 2: Head Automata Head-automata models constitute a second important model type that satisfy the sibling-decomposition assumption (bigram sibling models are a special case of head automata). These models make use of functions $g_R(i, s, s', r)$ where $s \in S, s' \in S$ are variables in a set of possible states S, and r is an index of a word in the sentence such that $i < r \leq n$. The function g_R returns a cost for taking word r as the next dependency, and transitioning from state s to s'. A similar function g_L is defined for left modifiers. We define

$$f_i(y_{|i}, s_0 \ldots s_q, t_0 \ldots t_p) =$$
$$\sum_{k=1}^{q} g_R(i, s_{k-1}, s_k, r_k) + \sum_{k=1}^{p} g_L(i, t_{k-1}, t_k, l_l)$$

to be the joint score for dependencies $y_{|i}$, and left and right state sequences $s_0 \ldots s_q$ and $t_0 \ldots t_p$. We specify that $s_0 = t_0 = \text{START}$ and $s_q = t_p = \text{END}$. In this case we define

$$f_i(y_{|i}) = \max_{s_0 \ldots s_q, t_0 \ldots t_p} f_i(y_{|i}, s_0 \ldots s_q, t_0 \ldots t_p)$$

and it follows that $\operatorname{argmax}_{y_{|i} \in \mathcal{Z}_i} f_i(y_{|i})$ can be computed in $O(n|S|^2)$ time using a variant of the Viterbi algorithm, hence the model satisfies the sibling-decomposition assumption. □

4 The Parsing Algorithm

We now describe the dual decomposition parsing algorithm for models that satisfy Assumption 1. Consider the following generalization of the decoding

```
Set $u^{(1)}(i,j) \leftarrow 0$ for all $(i,j) \in \mathcal{I}$
for $k = 1$ to $K$ do
    $y^{(k)} \leftarrow \operatorname*{argmax}_{y \in \mathcal{Y}} \sum_{(i,j) \in \mathcal{I}} \big(\gamma(i,j) + u^{(k)}(i,j)\big) y(i,j)$
    for $i \in \{0 \dots n\}$,
        $z^{(k)}_{|i} \leftarrow \operatorname*{argmax}_{z_{|i} \in \mathcal{Z}_i} \; (f_i(z_{|i}) - \sum_j u^{(k)}(i,j) z(i,j))$
    if $y^{(k)}(i,j) = z^{(k)}(i,j)$ for all $(i,j) \in \mathcal{I}$ then
        return $(y^{(k)}, z^{(k)})$
    for all $(i,j) \in \mathcal{I}$,
        $u^{(k+1)}(i,j) \leftarrow u^{(k)}(i,j) + \alpha_k (z^{(k)}(i,j) - y^{(k)}(i,j))$
return $(y^{(K)}, z^{(K)})$
```

Figure 1: The parsing algorithm for sibling decomposable models. $\alpha_k \geq 0$ for $k = 1 \dots K$ are step sizes, see Appendix A for details.

problem from Eq. 1, where $f(y) = \sum_i f_i(y_{|i})$, $h(y) = \sum_{(i,j) \in \mathcal{I}} \gamma(i,j) y(i,j)$, and $\gamma(i,j) \in \mathbb{R}$ for all (i,j):[1]

$$\operatorname*{argmax}_{z \in \mathcal{Z}, y \in \mathcal{Y}} \quad f(z) + h(y) \tag{4}$$

$$\text{such that} \quad z(i,j) = y(i,j) \text{ for all } (i,j) \in \mathcal{I} \tag{5}$$

Although the maximization w.r.t. z is taken over the set $\mathcal{Z}$, the constraints in Eq. 5 ensure that $z = y$ for some $y \in \mathcal{Y}$, and hence that $z \in \mathcal{Y}$.

Without the $z(i,j) = y(i,j)$ constraints, the objective would decompose into the separate maximizations $z^* = \operatorname{argmax}_{z \in \mathcal{Z}} f(z)$, and $y^* = \operatorname{argmax}_{y \in \mathcal{Y}} h(y)$, which can be easily solved using dynamic programming and MST, respectively. Thus, it is these constraints that complicate the optimization. Our approach gets around this difficulty by introducing new variables, $u(i,j)$, that serve to enforce agreement between the $y(i,j)$ and $z(i,j)$ variables. In the next section we will show that these $u(i,j)$ variables are actually Lagrange multipliers for the $z(i,j) = y(i,j)$ constraints.

Our parsing algorithm is shown in Figure 1. At each iteration k, the algorithm finds $y^{(k)} \in \mathcal{Y}$ using an MST algorithm, and $z^{(k)} \in \mathcal{Z}$ through separate decoding of the $(n+1)$ sibling models. The $u^{(k)}$ variables are updated if $y^{(k)}(i,j) \neq z^{(k)}(i,j)$ for some (i,j); these updates modify the objective functions for the two decoding steps, and intuitively encourage the $y^{(k)}$ and $z^{(k)}$ variables to be equal.

4.1 Lagrangian Relaxation

Recall that the main difficulty in solving Eq. 4 was the $z = y$ constraints. We deal with these constraints using *Lagrangian relaxation* (Lemaréchal, 2001). We first introduce Lagrange multipliers $u = \{u(i,j) : (i,j) \in \mathcal{I}\}$, and define the Lagrangian

$$\begin{aligned} L(u,y,z) \; &= \\ & f(z) + h(y) + \sum_{(i,j) \in \mathcal{I}} u(i,j) \Big(y(i,j) - z(i,j) \Big) \end{aligned} \tag{6}$$

If L^* is the optimal value of Eq. 4 subject to the constraints in Eq. 5, then for any value of u,

$$L^* = \max_{z \in \mathcal{Z}, y \in \mathcal{Y}, y = z} L(u,y,z) \tag{7}$$

This follows because if $y = z$, the right term in Eq. 6 is zero for any value of u. The dual objective $L(u)$ is obtained by omitting the $y = z$ constraint:

$$\begin{aligned} L(u) \; &= \; \max_{z \in \mathcal{Z}, y \in \mathcal{Y}} L(u,y,z) \\ &= \; \max_{z \in \mathcal{Z}} \Big(f(z) - \sum_{i,j} u(i,j) z(i,j) \Big) \\ & \quad + \max_{y \in \mathcal{Y}} \Big(h(y) + \sum_{i,j} u(i,j) y(i,j) \Big). \end{aligned}$$

Since $L(u)$ maximizes over a larger space (y may not equal z), we have that $L^* \leq L(u)$ (compare this to Eq. 7). The *dual problem*, which our algorithm optimizes, is to obtain the tightest such upper bound,

$$\text{(Dual problem)} \qquad \min_{u \in \mathbb{R}^{|\mathcal{I}|}} L(u). \tag{8}$$

The dual objective $L(u)$ is convex, but not differentiable. However, we can use a subgradient method to derive an algorithm that is similar to gradient descent, and which minimizes $L(u)$. A subgradient of a convex function $L(u)$ at u is a vector d_u such that for all $v \in \mathbb{R}^{|\mathcal{I}|}$, $L(v) \geq L(u) + d_u \cdot (v - u)$. By standard results,

$$d_{u^{(k)}} = y^{(k)} - z^{(k)}$$

is a subgradient for $L(u)$ at $u = u^{(k)}$, where $z^{(k)} = \operatorname{argmax}_{z \in \mathcal{Z}} f(z) - \sum_{i,j} u^{(k)}(i,j) z(i,j)$ and $y^{(k)} =$

[1] This is equivalent to Eq. 1 when $\gamma(i,j) = 0$ for all (i,j). In some cases, however, it is convenient to have a model with non-zero values for the γ variables; see the Appendix. Note that this definition of $h(y)$ allows $\operatorname{argmax}_{y \in \mathcal{Y}} h(y)$ to be calculated efficiently, using MST inference.

$\text{argmax}_{y \in \mathcal{Y}} h(y) + \sum_{i,j} u^{(k)}(i,j)y(i,j)$. Subgradient optimization methods are iterative algorithms with updates that are similar to gradient descent:

$$u^{(k+1)} = u^{(k)} - \alpha_k d_{u^{(k)}} = u^{(k)} - \alpha_k(y^{(k)} - z^{(k)}),$$

where α_k is a step size. It is easily verified that the algorithm in Figure 1 uses precisely these updates.

4.2 Formal Guarantees

With an appropriate choice of the step sizes α_k, the subgradient method can be shown to solve the dual problem, i.e.

$$\lim_{k \to \infty} L(u^{(k)}) = \min_u L(u).$$

See Korte and Vygen (2008), page 120, for details.

As mentioned before, the dual provides an upper bound on the optimum of the primal problem (Eq. 4),

$$\max_{z \in \mathcal{Z}, y \in \mathcal{Y}, y=z} f(z) + h(y) \leq \min_{u \in \mathbb{R}^{|\mathcal{I}|}} L(u). \quad (9)$$

However, we do not necessarily have strong duality—i.e., equality in the above equation—because the sets $\mathcal{Z}$ and $\mathcal{Y}$ are discrete sets. That said, for some functions $h(y)$ and $f(z)$ strong duality does hold, as stated in the following:

Theorem 1 *If for some $k \in \{1 \ldots K\}$ in the algorithm in Figure 1, $y^{(k)}(i,j) = z^{(k)}(i,j)$ for all $(i,j) \in \mathcal{I}$, then $(y^{(k)}, z^{(k)})$ is a solution to the maximization problem in Eq. 4.*

Proof. We have that $f(z^{(k)}) + h(y^{(k)}) = L(u^{(k)}, z^{(k)}, y^{(k)}) = L(u^{(k)})$, where the last equality is because $y^{(k)}$, $z^{(k)}$ are defined as the respective argmax's. Thus, the inequality in Eq. 9 is tight, and ($y^{(k)}$, $z^{(k)}$) and $u^{(k)}$ are primal and dual optimal. □

Although the algorithm is not guaranteed to satisfy $y^{(k)} = z^{(k)}$ for some k, by Theorem 1 if it does reach such a state, then we have the guarantee of an *exact* solution to Eq. 4, with the dual solution u providing a certificate of optimality. We show in the experiments that this occurs very frequently, in spite of the parsing problem being NP-hard.

It can be shown that Eq. 8 is the dual of an LP relaxation of the original problem. When the conditions of Theorem 1 are satisfied, it means that the LP relaxation is *tight* for this instance. For brevity we omit the details, except to note that when the LP relaxation is *not* tight, the optimal primal solution to the LP relaxation could be recovered by averaging methods (Nedić and Ozdaglar, 2009).

5 Grandparent Dependency Models

In this section we extend the approach to consider grandparent relations. In grandparent models each parse tree y is represented as a vector

$$y = \{y(i,j) : (i,j) \in \mathcal{I}\} \cup \{y_\uparrow(i,j) : (i,j) \in \mathcal{I}\}$$

where we have added a second set of duplicate variables, $y_\uparrow(i,j)$ for all $(i,j) \in \mathcal{I}$. The set of all valid parse trees is then defined as

$$\begin{aligned} \mathcal{Y} = \{y \ : \ & y(i,j) \text{ variables form a directed tree,} \\ & y_\uparrow(i,j) = y(i,j) \text{ for all } (i,j) \in \mathcal{I}\} \end{aligned}$$

We again partition the variables into $n+1$ subsets, $y_{|0} \ldots y_{|n}$, by (re)defining

$$\begin{aligned} y_{|i} \ = \ & \{y(i,j) : j = 1 \ldots n, j \neq i\} \\ & \cup \{y_\uparrow(k,i) : k = 0 \ldots n, k \neq i\} \end{aligned}$$

So as before $y_{|i}$ contains variables $y(i,j)$ which indicate which words modify the i'th word. In addition, $y_{|i}$ includes $y_\uparrow(k,i)$ variables that indicate the word that word i itself modifies.

The set of all possible values of $y_{|i}$ is now

$$\begin{aligned} \mathcal{Z}_i = \{y_{|i} : \ & y(i,j) \in \{0,1\} \text{ for } j = 1 \ldots n, j \neq i; \\ & y_\uparrow(k,i) \in \{0,1\} \text{ for } k = 0 \ldots n, k \neq i; \\ & \sum_k y_\uparrow(k,i) = 1\} \end{aligned}$$

Hence the $y(i,j)$ variables can take any values, but only one of the $y_\uparrow(k,i)$ variables can be equal to 1 (as only one word can be a parent of word i). As before, we define $\mathcal{Z} = \{y : y_{|i} \in \mathcal{Z}_i \text{ for } i = 0 \ldots n\}$.

We introduce the following assumption:

Assumption 2 (GS Decompositions)
A model $f(y)$ satisfies the grandparent/sibling-decomposition (GSD) assumption if: 1) $f(z) = \sum_{i=0}^n f_i(z_{|i})$ for some set of functions $f_0 \ldots f_n$. 2) For any $i \in \{0 \ldots n\}$, for any value of the variables $u(i,j) \in \mathbb{R}$ for $j = 1 \ldots n$, and $v(k,i) \in \mathbb{R}$ for $k = 0 \ldots n$, it is possible to compute

$$\underset{z_{|i} \in \mathcal{Z}_i}{\text{argmax}} (f_i(z_{|i}) - \sum_j u(i,j)z(i,j) - \sum_k v(k,i)z_\uparrow(k,i))$$

in polynomial time.

Again, it follows that we can approximate $y^* = \text{argmax}_{y\in\mathcal{Y}} \sum_{i=0}^{n} f_i(y_{|i})$ by $z^* = \text{argmax}_{z\in\mathcal{Z}} \sum_{i=0}^{n} f_i(z_{|i})$, by defining $z^*_{|i} = \text{argmax}_{z_{|i}\in\mathcal{Z}_i} f_i(z_{|i})$ for $i = 0\ldots n$. The resulting vector z^* may be deficient in two respects. First, the variables $z^*(i,j)$ may not form a well-formed directed spanning tree. Second, we may have $z^*_{\uparrow}(i,j) \neq z^*(i,j)$ for some values of (i,j).

Example 3: Grandparent/Sibling Models An important class of models that satisfy Assumption 2 are defined as follows. Again, for a vector $y_{|i}$ define $l_1 \ldots l_p$ to be the sequence of left modifiers to word i under $y_{|i}$, and $r_1 \ldots r_q$ to be the set of right modifiers. Define k^* to the value for k such that $y_{\uparrow}(k,i) = 1$. Then the model is defined as follows:

$$f_i(y_{|i}) = \sum_{j=1}^{p+1} g_L(i, k^*, l_{j-1}, l_j) + \sum_{j=1}^{q+1} g_R(i, k^*, r_{j-1}, r_j)$$

This is very similar to the bigram-sibling model, but with the modification that the g_L and g_R functions depend in addition on the value for k^*. This allows these functions to model grandparent dependencies such as (k^*, i, l_j) and sibling dependencies such as (i, l_{j-1}, l_j). Finding $z^*_{|i}$ under the definition can be accomplished in $O(n^3)$ time, by decoding the model using dynamic programming separately for each of the $O(n)$ possible values of k^*, and picking the value for k^* that gives the maximum value under these decodings. □

A dual-decomposition algorithm for models that satisfy the GSD assumption is shown in Figure 2. The algorithm can be justified as an instance of Lagrangian relaxation applied to the problem

$$\underset{z\in\mathcal{Z}, y\in\mathcal{Y}}{\text{argmax}} \quad f(z) + h(y) \tag{10}$$

with constraints

$$z(i,j) = y(i,j) \;\text{ for all } (i,j) \in \mathcal{I} \tag{11}$$

$$z_{\uparrow}(i,j) = y(i,j) \;\text{ for all } (i,j) \in \mathcal{I} \tag{12}$$

The algorithm employs two sets of Lagrange multipliers, $u(i,j)$ and $v(i,j)$, corresponding to constraints in Eqs. 11 and 12. As in Theorem 1, if at any point in the algorithm $z^{(k)} = y^{(k)}$, then $(z^{(k)}, y^{(k)})$ is an exact solution to the problem in Eq. 10.

```
Set $u^{(1)}(i,j) \leftarrow 0$, $v^{(1)}(i,j) \leftarrow 0$ for all $(i,j) \in \mathcal{I}$
for $k = 1$ to $K$ do
    $y^{(k)} \leftarrow \text{argmax}_{y\in\mathcal{Y}} \sum_{(i,j)\in\mathcal{I}} y(i,j)\theta(i,j)$
    where $\theta(i,j) = \gamma(i,j) + u^{(k)}(i,j) + v^{(k)}(i,j)$.
    for $i \in \{0 \ldots n\}$,
        $z^{(k)}_{|i} \leftarrow \text{argmax}_{z_{|i}\in\mathcal{Z}_i} (f_i(z_{|i}) - \sum_j u^{(k)}(i,j)z(i,j)$
                                    $- \sum_j v^{(k)}(j,i)z_{\uparrow}(j,i))$
    if $y^{(k)}(i,j) = z^{(k)}(i,j) = z^{(k)}_{\uparrow}(i,j)$ for all $(i,j) \in \mathcal{I}$
    then
        return $(y^{(k)}, z^{(k)})$
    for all $(i,j) \in \mathcal{I}$,
        $u^{(k+1)}(i,j) \leftarrow u^{(k)}(i,j) + \alpha_k(z^{(k)}(i,j) - y^{(k)}(i,j))$
        $v^{(k+1)}(i,j) \leftarrow v^{(k)}(i,j) + \alpha_k(z^{(k)}_{\uparrow}(i,j) - y^{(k)}(i,j))$
return $(y^{(K)}, z^{(K)})$
```

Figure 2: The parsing algorithm for grandparent/sibling-decomposable models.

6 The Training Algorithm

In our experiments we make use of discriminative linear models, where for an input sentence x, the score for a parse y is $f(y) = w \cdot \phi(x,y)$ where $w \in \mathbb{R}^d$ is a parameter vector, and $\phi(x,y) \in \mathbb{R}^d$ is a feature-vector representing parse tree y in conjunction with sentence x. We will assume that the features decompose in the same way as the sibling-decomposable or grandparent/sibling-decomposable models, that is $\phi(x,y) = \sum_{i=0}^{n} \phi(x, y_{|i})$ for some feature vector definition $\phi(x, y_{|i})$. In the *bigram sibling* models in our experiments, we assume that

$$\phi(x, y_{|i}) = \sum_{k=1}^{p+1} \phi_L(x, i, l_{k-1}, l_k) + \sum_{k=1}^{q+1} \phi_R(x, i, r_{k-1}, r_k)$$

where as before $l_1 \ldots l_p$ and $r_1 \ldots r_q$ are left and right modifiers under $y_{|i}$, and where ϕ_L and ϕ_R are feature vector definitions. In the *grandparent models* in our experiments, we use a similar definition with feature vectors $\phi_L(x, i, k^*, l_{k-1}, l_k)$ and $\phi_R(x, i, k^*, r_{k-1}, r_k)$, where k^* is the parent for word i under $y_{|i}$.

We train the model using the averaged perceptron for structured problems (Collins, 2002). Given the i'th example in the training set, $(x^{(i)}, y^{(i)})$, the perceptron updates are as follows:

- $z^* = \text{argmax}_{y\in\mathcal{Z}}\, w \cdot \phi(x^{(i)}, y)$
- If $z^* \neq y^{(i)}$, $w = w + \phi(x^{(i)}, y^{(i)}) - \phi(x^{(i)}, z^*)$

The first step involves inference over the set $\mathcal{Z}$, rather than $\mathcal{Y}$ as would be standard in the perceptron. Thus, decoding during training can be achieved by dynamic programming over head automata alone, which is very efficient.

Our training approach is closely related to *local training methods* (Punyakanok et al., 2005). We have found this method to be effective, very likely because $\mathcal{Z}$ is a superset of $\mathcal{Y}$. Our training algorithm is also related to recent work on training using *outer bounds* (see, e.g., (Taskar et al., 2003; Finley and Joachims, 2008; Kulesza and Pereira, 2008; Martins et al., 2009)). Note, however, that the LP relaxation optimized by dual decomposition is significantly tighter than $\mathcal{Z}$. Thus, an alternative approach would be to use the dual decomposition algorithm for inference during training.

7 Experiments

We report results on a number of data sets. For comparison to Martins et al. (2009), we perform experiments for Danish, Dutch, Portuguese, Slovene, Swedish and Turkish data from the CoNLL-X shared task (Buchholz and Marsi, 2006), and English data from the CoNLL-2008 shared task (Surdeanu et al., 2008). We use the official training/test splits for these data sets, and the same evaluation methodology as Martins et al. (2009). For comparison to Smith and Eisner (2008), we also report results on Danish and Dutch using their alternate training/test split. Finally, we report results on the English WSJ treebank, and the Prague treebank. We use feature sets that are very similar to those described in Carreras (2007). We use marginal-based pruning, using marginals calculated from an arc-factored spanning tree model using the matrix-tree theorem (McDonald and Satta, 2007; Smith and Smith, 2007; Koo et al., 2007).

In all of our experiments we set the value K, the maximum number of iterations of dual decomposition in Figures 1 and 2, to be 5,000. If the algorithm does not terminate—i.e., it does not return $(y^{(k)}, z^{(k)})$ within 5,000 iterations—we simply take the parse $y^{(k)}$ with the maximum value of $f(y^{(k)})$ as the output from the algorithm. At first sight 5,000 might appear to be a large number, but decoding is still fast—see Sections 7.3 and 7.4 for discussion.[2] The strategy for choosing step sizes α_k is described in Appendix A, along with other details.

We first discuss performance in terms of *accuracy*, *success in recovering an exact solution*, and *parsing speed*. We then describe additional experiments examining various aspects of the algorithm.

7.1 Accuracy

Table 1 shows results for previous work on the various data sets, and results for an arc-factored model with pure MST decoding with our features. (We use the acronym UAS (unlabeled attachment score) for dependency accuracy.) We also show results for the bigram-sibling and grandparent/sibling (G+S) models under dual decomposition. Both the bigram-sibling and G+S models show large improvements over the arc-factored approach; they also compare favorably to previous work—for example the G+S model gives better results than all results reported in the CoNLL-X shared task, on all languages. Note that we use different feature sets from both Martins et al. (2009) and Smith and Eisner (2008).

7.2 Success in Recovering Exact Solutions

Next, we consider how often our algorithms return an exact solution to the original optimization problem, with a certificate—i.e., how often the algorithms in Figures 1 and 2 terminate with $y^{(k)} = z^{(k)}$ for some value of $k < 5000$ (and are thus optimal, by Theorem 1). The CertS and CertG columns in Table 1 give the results for the sibling and G+S models respectively. For all but one setting[3] over 95% of the test sentences are decoded exactly, with 99% exactness in many cases.

For comparison, we also ran both the single-commodity flow and multiple-commodity flow LP relaxations of Martins et al. (2009) with our models and features. We measure how often these relaxations terminate with an exact solution. The results in Table 2 show that our method gives exact solutions more often than both of these relaxations.[4] In computing the accuracy figures for Martins et al.

[2] Note also that the feature vectors ϕ and inner products $w \cdot \phi$ only need to be computed once, thus saving computation.

[3] The exception is Slovene, which has the smallest training set at only 1534 sentences.

[4] Note, however, that it is possible that the Martins et al. relaxations would have given a higher proportion of integral solutions if their relaxation was used during training.

	Ma09	MST	Sib	G+S	Best	CertS	CertG	TimeS	TimeG	TrainS	TrainG
Dan	91.18	89.74	91.08	**91.78**	91.54	99.07	98.45	0.053	0.169	0.051	0.109
Dut	85.57	82.33	84.81	**85.81**	85.57	98.19	97.93	0.035	0.120	0.046	0.048
Por	92.11	90.68	92.57	**93.03**	92.11	99.65	99.31	0.047	0.257	0.077	0.103
Slo	85.61	82.39	84.89	**86.21**	85.61	90.55	95.27	0.158	0.295	0.054	0.130
Swe	90.60	88.79	90.10	**91.36**	90.60	98.71	98.97	0.035	0.141	0.036	0.055
Tur	76.34	75.66	77.14	**77.55**	76.36	98.72	99.04	0.021	0.047	0.016	0.036
Eng^1	91.16	89.20	91.18	**91.59**	—	98.65	99.18	0.082	0.200	0.032	0.076
Eng^2	—	90.29	92.03	**92.57**	—	98.96	99.12	0.081	0.168	0.032	0.076

	Sm08	MST	Sib	G+S	—	CertS	CertG	TimeS	TimeG	TrainS	TrainG
Dan	86.5	87.89	89.58	**91.00**	—	98.50	98.50	0.043	0.120	0.053	0.065
Dut	88.5	88.86	90.87	**91.76**	—	98.00	99.50	0.036	0.046	0.050	0.054

	Mc06	MST	Sib	G+S	—	CertS	CertG	TimeS	TimeG	TrainS	TrainG
PTB	91.5	90.10	91.96	**92.46**	—	98.89	98.63	0.062	0.210	0.028	0.078
PDT	85.2	84.36	86.44	**87.32**	—	96.67	96.43	0.063	0.221	0.019	0.051

Table 1: A comparison of non-projective automaton-based parsers with results from previous work. MST: Our first-order baseline. Sib/G+S: Non-projective head automata with sibling or grandparent/sibling interactions, decoded via dual decomposition. Ma09: The best UAS of the LP/ILP-based parsers introduced in Martins et al. (2009). Sm08: The best UAS of any LBP-based parser in Smith and Eisner (2008). Mc06: The best UAS reported by McDonald and Pereira (2006). Best: For the CoNLL-X languages only, the best UAS for any parser in the original shared task (Buchholz and Marsi, 2006) or in any column of Martins et al. (2009, Table 1); note that the latter includes McDonald and Pereira (2006), Nivre and McDonald (2008), and Martins et al. (2008). CertS/CertG: Percent of test examples for which dual decomposition produced a certificate of optimality, for Sib/G+S. TimeS/TimeG: Seconds/sentence for test decoding, for Sib/G+S. TrainS/TrainG: Seconds/sentence during training, for Sib/G+S. For consistency of timing, test decoding was carried out on identical machines with zero additional load; however, training was conducted on machines with varying hardware and load. We ran two tests on the CoNLL-08 corpus. Eng^1: UAS when testing on the CoNLL-08 validation set, following Martins et al. (2009). Eng^2: UAS when testing on the CoNLL-08 test set.

(2009), we project fractional solutions to a well-formed spanning tree, as described in that paper.

Finally, to better compare the tightness of our LP relaxation to that of earlier work, we consider randomly-generated instances. Table 2 gives results for our model and the LP relaxations of Martins et al. (2009) with randomly generated scores on automata transitions. We again recover exact solutions more often than the Martins et al. relaxations. Note that with random parameters the percentage of exact solutions is significantly lower, suggesting that the exactness of decoding of the trained models is a special case. We speculate that this is due to the high performance of approximate decoding with $\mathcal{Z}$ in place of $\mathcal{Y}$ under the trained models for f_i; the training algorithm described in section 6 may have the tendency to make the LP relaxation tight.

7.3 Speed

Table 1, columns TimeS and TimeG, shows decoding times for the dual decomposition algorithms. Table 2 gives speed comparisons to Martins et al. (2009). Our method gives significant speed-ups over the Martins et al. (2009) method, presumably because it leverages the underlying structure of the problem, rather than using a generic solver.

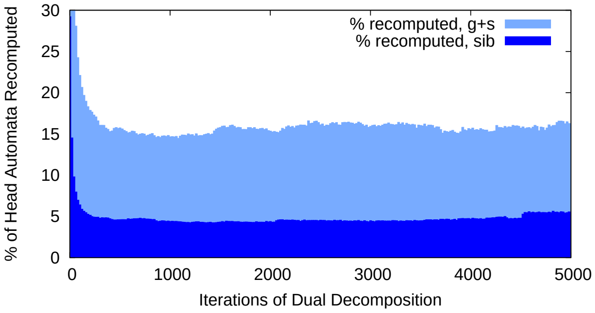


Figure 3: The average percentage of head automata that must be recomputed on each iteration of dual decomposition on the PTB validation set.

7.4 Lazy Decoding

Here we describe an important optimization in the dual decomposition algorithms. Consider the algorithm in Figure 1. At each iteration we must find

$$z_{|i}^{(k)} = \operatorname*{argmax}_{z_{|i} \in \mathcal{Z}_i} (f_i(z_{|i}) - \sum_j u^{(k)}(i,j) z(i,j))$$

Sib	Acc	Int	Time	Rand
LP(S)	92.14	88.29	0.14	11.7
LP(M)	92.17	93.18	0.58	30.6
ILP	92.19	100.0	1.44	100.0
DD-5000	92.19	98.82	0.08	35.6
DD-250	92.23	89.29	0.03	10.2

G+S	Acc	Int	Time	Rand
LP(S)	92.60	91.64	0.23	0.0
LP(M)	92.58	94.41	0.75	0.0
ILP	92.70	100.0	1.79	100.0
DD-5000	92.71	98.76	0.23	6.8
DD-250	92.66	85.47	0.12	0.0

Table 2: A comparison of dual decomposition with linear programs described by Martins et al. (2009). LP(S): Linear Program relaxation based on single-commodity flow. LP(M): Linear Program relaxation based on multi-commodity flow. ILP: Exact Integer Linear Program. DD-5000/DD-250: Dual decomposition with non-projective head automata, with $K = 5000/250$. Upper results are for the sibling model, lower results are G+S. Columns give scores for UAS accuracy, percentage of solutions which are integral, and solution speed in seconds per sentence. These results are for Section 22 of the PTB. The last column is the percentage of integral solutions on a random problem of length 10 words. The (I)LP experiments were carried out using Gurobi, a high-performance commercial-grade solver.

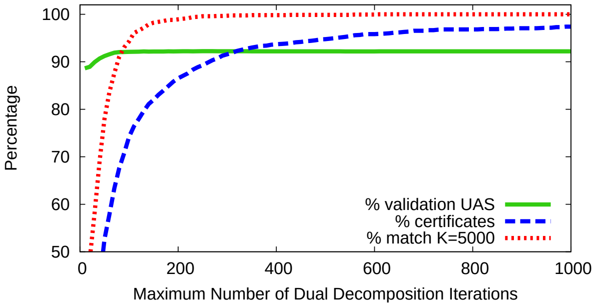


Figure 4: The behavior of the dual-decomposition parser with sibling automata as the value of K is varied.

	Sib	P-Sib	G+S	P-G+S
PTB	92.19	92.34	92.71	92.70
PDT	86.41	85.67	87.40	86.43

Table 3: UAS of projective and non-projective decoding for the English (PTB) and Czech (PDT) validation sets. Sib/G+S: as in Table 1. P-Sib/P-G+S: Projective versions of Sib/G+S, where the MST component has been replaced with the Eisner (2000) first-order projective parser.

for $i = 0 \ldots n$. However, if for some i, $u^{(k)}(i,j) = u^{(k-1)}(i,j)$ for all j, then $z^{(k)}_{|i} = z^{(k-1)}_{|i}$. In *lazy decoding* we immediately set $z^{(k)}_{|i} = z^{(k-1)}_{|i}$ if $u^{(k)}(i,j) = u^{(k-1)}(i,j)$ for all j; this check takes $O(n)$ time, and saves us from decoding with the i'th automaton. In practice, the updates to u are very sparse, and this condition occurs very often in practice. Figure 3 demonstrates the utility of this method for both sibling automata and G+S automata.

7.5 Early Stopping

We also ran experiments varying the value of K—the maximum number of iterations—in the dual decomposition algorithms. As before, if we do not find $y^{(k)} = z^{(k)}$ for some value of $k \leq K$, we choose the $y^{(k)}$ with optimal value for $f(y^{(k)})$ as the final solution. Figure 4 shows three graphs: 1) the accuracy of the parser on PTB validation data versus the value for K; 2) the percentage of examples where $y^{(k)} = z^{(k)}$ at some point during the algorithm, hence the algorithm returns a certificate of optimality; 3) the percentage of examples where the solution returned is the same as the solution for the algorithm with $K = 5000$ (our original setting). It can be seen for K as small as 250 we get very similar accuracy to $K = 5000$ (see Table 2). In fact, for this setting the algorithm returns the same solution as for $K = 5000$ on 99.59% of the examples. However only 89.29% of these solutions are produced with a certificate of optimality ($y^{(k)} = z^{(k)}$).

7.6 How Good is the Approximation z^*?

We ran experiments measuring the quality of $z^* = \text{argmax}_{z \in \mathcal{Z}} f(z)$, where $f(z)$ is given by the perceptron-trained bigram-sibling model. Because z^* may not be a well-formed tree with n dependencies, we report precision and recall rather than conventional dependency accuracy. Results on the PTB validation set were 91.11%/88.95% precision/recall, which is accurate considering the unconstrained nature of the predictions. Thus the z^* approximation is clearly a good one; we suspect that this is one reason for the good convergence results for the method.

7.7 Importance of Non-Projective Decoding

It is simple to adapt the dual-decomposition algorithms in figures 1 and 2 to give *projective* dependency structures: the set $\mathcal{Y}$ is redefined to be the set of all projective structures, with the $\arg\max$ over $\mathcal{Y}$ being calculated using a projective first-order parser (Eisner, 2000). Table 3 shows results for projective and non-projective parsing using the dual decomposition approach. For Czech data, where non-projective structures are common, non-projective decoding has clear benefits. In contrast, there is little difference in accuracy between projective and non-projective decoding on English.

8 Conclusions

We have described dual decomposition algorithms for non-projective parsing, which leverage existing dynamic programming and MST algorithms. There are a number of possible areas for future work. As described in section 7.7, the algorithms can be easily modified to consider projective structures by replacing $\mathcal{Y}$ with the set of projective trees, and then using first-order dependency parsing algorithms in place of MST decoding. This method could be used to derive parsing algorithms that include higher-order features, as an alternative to specialized dynamic programming algorithms. Eisner (2000) describes extensions of head automata to include word senses; we have not discussed this issue in the current paper, but it is simple to develop dual decomposition algorithms for this case, using similar methods to those used for the grandparent models. The general approach should be applicable to other lexicalized syntactic formalisms, and potentially also to decoding in syntax-driven translation. In addition, our dual decomposition approach is well-suited to parallelization. For example, each of the head-automata could be optimized independently in a multi-core or GPU architecture. Finally, our approach could be used with other structured learning algorithms, e.g. Meshi et al. (2010).

A Implementation Details

This appendix describes details of the algorithm, specifically choice of the step sizes α_k, and use of the $\gamma(i,j)$ parameters.

A.1 Choice of Step Sizes

We have found the following method to be effective. First, define $\delta = f(z^{(1)}) - f(y^{(1)})$, where $(z^{(1)}, y^{(1)})$ is the output of the algorithm on the first iteration (note that we always have $\delta \geq 0$ since $f(z^{(1)}) = L(u^{(1)})$). Then define $\alpha_k = \delta/(1 + \eta_k)$, where η_k is the number of times that $L(u^{(k')}) > L(u^{(k'-1)})$ for $k' \leq k$. Hence the learning rate drops at a rate of $1/(1+t)$, where t is the number of times that the dual increases from one iteration to the next.

A.2 Use of the $\gamma(i,j)$ Parameters

The parsing algorithms both consider a generalized problem that includes $\gamma(i,j)$ parameters. We now describe how these can be useful. Recall that the optimization problem is to solve $\mathrm{argmax}_{z\in\mathcal{Z},y\in\mathcal{Y}}\, f(z) + h(y)$, subject to a set of agreement constraints. In our models, $f(z)$ can be written as $f'(z) + \sum_{i,j} \alpha(i,j)z(i,j)$ where $f'(z)$ includes only terms depending on higher-order (non arc-factored features), and $\alpha(i,j)$ are weights that consider the dependency between i and j alone. For any value of $0 \leq \beta \leq 1$, the problem $\mathrm{argmax}_{z\in\mathcal{Z},y\in\mathcal{Y}}\, f_2(z) + h_2(y)$ is equivalent to the original problem, if $f_2(z) = f'(z) + (1-\beta)\sum_{i,j} \alpha(i,j)z(i,j)$ and $h_2(y) = \beta\sum_{i,j} \alpha(i,j)y(i,j)$. We have simply shifted the $\alpha(i,j)$ weights from one model to the other. While the optimization problem remains the same, the algorithms in Figure 1 and 2 will converge at different rates depending on the value for β. In our experiments we set $\beta = 0.001$, which puts almost all the weight in the head-automata models, but allows weights on spanning tree edges to break ties in MST inference in a sensible way. We suspect this is important in early iterations of the algorithm, when many values for $u(i,j)$ or $v(i,j)$ will be zero, and where with $\beta = 0$ many spanning tree solutions $y^{(k)}$ would be essentially random, leading to very noisy updates to the $u(i,j)$ and $v(i,j)$ values. We have not tested other values for β.

Acknowledgments MIT gratefully acknowledges the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the DARPA, AFRL, or the US government. A. Rush was supported by the GALE program of the DARPA, Contract No. HR0011-06-C-0022. D. Sontag was supported by a Google PhD Fellowship.

References

H. Alshawi. 1996. Head Automata and Bilingual Tiling: Translation with Minimal Representations. In *Proc. ACL*, pages 167–176.

S. Buchholz and E. Marsi. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *Proc. CoNLL*, pages 149–164.

X. Carreras. 2007. Experiments with a Higher-Order Projective Dependency Parser. In *Proc. EMNLP-CoNLL*, pages 957–961.

M. Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proc. EMNLP*, pages 1–8.

J. Duchi, D. Tarlow, G. Elidan, and D. Koller. 2007. Using Combinatorial Optimization within Max-Product Belief Propagation. In *NIPS*, pages 369–376.

J. Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62.

T. Finley and T. Joachims. 2008. Training structural svms when exact inference is intractable. In *ICML*, pages 304–311.

A.K. Joshi and Y. Schabes. 1997. Tree-Adjoining Grammars. *Handbook of Formal Languages: Beyond Words*, 3:69–123.

N. Komodakis, N. Paragios, and G. Tziritas. 2007. MRF Optimization via Dual Decomposition: Message-Passing Revisited. In *Proc. ICCV*.

T. Koo, A. Globerson, X. Carreras, and M. Collins. 2007. Structured Prediction Models via the Matrix-Tree Theorem. In *Proc. EMNLP-CoNLL*, pages 141–150.

B.H. Korte and J. Vygen. 2008. *Combinatorial Optimization: Theory and Algorithms*. Springer Verlag.

A. Kulesza and F. Pereira. 2008. Structured learning with approximate inference. In *NIPS*.

C. Lemaréchal. 2001. Lagrangian Relaxation. In *Computational Combinatorial Optimization, Optimal or Provably Near-Optimal Solutions [based on a Spring School]*, pages 112–156, London, UK. Springer-Verlag.

A.F.T. Martins, D. Das, N.A. Smith, and E.P. Xing. 2008. Stacking Dependency Parsers. In *Proc. EMNLP*, pages 157–166.

A.F.T. Martins, N.A. Smith., and E.P. Xing. 2009. Concise Integer Linear Programming Formulations for Dependency Parsing. In *Proc. ACL*, pages 342–350.

R. McDonald and F. Pereira. 2006. Online Learning of Approximate Dependency Parsing Algorithms. In *Proc. EACL*, pages 81–88.

R. McDonald and G. Satta. 2007. On the Complexity of Non-Projective Data-Driven Dependency Parsing. In *Proc. IWPT*.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proc. HLT-EMNLP*, pages 523–530.

O. Meshi, D. Sontag, T. Jaakkola, and A. Globerson. 2010. Learning Efficiently with Approximate Inference via Dual Losses. In *Proc. ICML*.

A. Nedić and A. Ozdaglar. 2009. Approximate Primal Solutions and Rate Analysis for Dual Subgradient Methods. *SIAM Journal on Optimization*, 19(4):1757–1780.

J. Nivre and R. McDonald. 2008. Integrating Graph-Based and Transition-Based Dependency Parsers. In *Proc. ACL*, pages 950–958.

V. Punyakanok, D. Roth, W. Yih, and D. Zimak. 2005. Learning and Inference over Constrained Output. In *Proc. IJCAI*, pages 1124–1129.

S. Riedel and J. Clarke. 2006. Incremental Integer Linear Programming for Non-projective Dependency Parsing. In *Proc. EMNLP*, pages 129–137.

A.M. Rush, D. Sontag, M. Collins, and T. Jaakkola. 2010. On Dual Decomposition and Linear Programming Relaxations for Natural Language Processing. In *Proc. EMNLP*.

D.A. Smith and J. Eisner. 2008. Dependency Parsing by Belief Propagation. In *Proc. EMNLP*, pages 145–156.

D.A. Smith and N.A. Smith. 2007. Probabilistic Models of Nonprojective Dependency Trees. In *Proc. EMNLP-CoNLL*, pages 132–140.

D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. 2008. Tightening LP Relaxations for MAP using Message Passing. In *Proc. UAI*.

M. Steedman. 2000. *The Syntactic Process*. MIT Press.

M. Surdeanu, R. Johansson, A. Meyers, L. Màrquez, and J. Nivre. 2008. The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In *Proc. CoNLL*.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin Markov networks. In *NIPS*.

M. Wainwright, T. Jaakkola, and A. Willsky. 2005. MAP estimation via agreement on trees: message-passing and linear programming. In *IEEE Transactions on Information Theory*, volume 51, pages 3697–3717.

Author Index

Agarwal, Apoorv, 1024
Aguiar, Pedro, 34
Ahmed, Amr, 1140
Aker, Ahmet, 482
Allauzen, Alexandre, 778, 933
Alshawi, Hiyan, 705
Ananthakrishnan, Sankaranarayanan, 626
Angeli, Gabor, 502

Baldridge, Jason, 196
Baldwin, Timothy, 694, 862
Baroni, Marco, 1183
Barzilay, Regina, 377, 853, 1234
Bederson, Benjamin B., 127
Ben Hamadou, Abdelmajid, 953
Benajiba, Yassine, 993
Bergsma, Shane, 293
Berleant, Shoshana, 564
Bhargava, Aditya, 293
Bienenstock, Elie, 799
Blanco, Eduardo, 315
Blunsom, Phil, 1204
Bodrumlu, Tugba, 524
Bollegala, Danushka, 399
Boudin, Florian, 108
Boxwell, Stephen, 736
Boyd-Graber, Jordan, 45, 284
Brew, Chris, 736
Brody, Samuel, 1214
Bu, Fan, 1119
Bunescu, Razvan, 97
Busby, George, 810
Buyko, Ekaterina, 982
Buzek, Olivia, 127
Byrne, William, 545

Cancedda, Nicola, 606
Cardie, Claire, 1046
Carenini, Giuseppe, 388
Cavedon, Lawrence, 862
Chai, Joyce, 471, 756
Chambers, Nate, 492
Chang, Baobao, 789
Chang, Ching-Yun, 1194
Chang, Ming-Wei, 767
Chang, Pi-Chuan, 705
Chang, Yi, 1129
Charniak, Eugene, 674
Chatterjee, Samidh, 606
Chen, Bin, 872
Chen, Harr, 1234
Cheung, Jackie Chi Kit, 23
Chiticariu, Laura, 1002
Chng, Eng Siong, 440
Cholakov, Kostadin, 902
Christodoulopoulos, Christos, 575
Chung, Tagyoung, 636
Church, Ken, 460
Clark, Stephen, 843, 1194
Cmejrek, Martin, 138
Cohn, Trevor, 482, 1204
Collins, Michael, 1, 1288
Connor, Michael, 767
Contractor, Danish, 87
Coppersmith, Glen, 460
Crammer, Koby, 971
Crisafulli, Giuseppe, 116

Dahlmeier, Daniel, 923
Dale, Robert, 913
Daume III, Hal, 77
Davis, Alan, 229
Davis, Jesse, 1088
Dawes, Martin, 108
de Gispert, Adrià, 545
Dhillon, Paramveer S., 725
Dinu, Georgiana, 1162

Do, Quang, 1099
Dong, Anlei, 1129
Dredze, Mark, 460, 585
Dridan, Rebecca, 694
Du, Jinhua, 420
Duh, Kevin, 944
Dyer, Chris, 534

Eidelman, Vladimir, 821
Eisenstein, Jacob, 1277
Erk, Katrin, 196
Espinosa, Dominic, 564
Etzioni, Oren, 1088, 1266

Faruquie, Tanveer, 87
Feng, Yansong, 513
Figueiredo, Mario, 34
Florian, Radu, 335
Foster, Dean, 725
Foster, George, 451

Gaizauskas, Robert, 482
Gargouri, Bilel, 953
Genzel, Dmitriy, 158
Gildea, Daniel, 596, 636
Girju, Roxana, 66
Goldwater, Sharon, 575, 1223
Golland, Dave, 410
Goutte, Cyril, 451
Goyal, Amit, 77
Greene, Erica, 524
Gurevych, Iryna, 1035
Guthrie, David, 262

Haertel, Robbie, 810
Haghighi, Aria, 377, 853
Hahn, Udo, 982
Han, Bo, 1068
Han, Dongxu, 789
Hao, Yu, 1119
Hardisty, Eric, 284
Harper, Mary, 12, 821
Hassan, Ahmed, 1245
He, Hua, 293
He, Zhongjun, 555
Heal, Kristian, 810
Hepple, Mark, 262

Hernault, Hugo, 399
Hirao, Tsutomu, 944
Hopkins, Mark, 646
Hovy, Eduard, 1110
Hu, Chang, 127
Huang, Liang, 273
Huang, Wenyi, 366
Huang, Xuanjing, 187
Huang, Yunfeng, 97
Huang, Zhongqiang, 12, 138, 821
Hwang, Seung-won, 430

Ishizuka, Mitsuru, 399
Isozaki, Hideki, 944

Jaakkola, Tommi, 1, 1288
Jakob, Niklas, 1035
Jansen, Aren, 460
Jiang, Jie, 420
Jiang, Jing, 56
Jiang, Long, 430
Jiang, Yunliang, 1077
Johnson, Mark, 1234
Joty, Shafiq, 388
Jurafsky, Dan, 492

Kan, Min-Yen, 148
Khemakhem, Aida, 953
Kim, Su Nam, 862
Klapaftis, Ioannis, 745
Klein, Dan, 410, 502
Knight, Kevin, 524
Kolari, Pranam, 1129
Konda, Yuki, 1129
Kondrak, Grzegorz, 293
Kong, Fang, 882
Koo, Terry, 1288
Kothari, Govind, 87
Kozareva, Zornitsa, 1110
Krishnamurthy, Rajasekar, 1002
Kronrod, Yakov, 127
Kuhn, Roland, 451
Kumar, Shaishav, 1256
Kwiatkowksi, Tom, 1223

Lamar, Michael, 799
Langmead, Greg, 646

Lapata, Mirella, 513, 1162
Le, Hai Son, 778
Lee, Heeyoung, 492
Lee, Yoong Keok, 853
Li, Haizhou, 440
Li, Junhui, 714
Li, Kuan, 1068
Li, Xiao-Li, 218
Li, Xiaoming, 56
Li, Yunyao, 1002
Liang, Percy, 410, 502
Liberman, Mark, 725
Lin, Cindy Xide, 1077
Lin, Thomas, 1266
Liu, Bing, 207, 218
Liu, Chang, 923
Liu, Xiaohua, 1068
Liu, Zhiyuan, 366
Lonsdale, Deryle, 810
Lu, Wei, 177
Lund, William B., 240
Luong, Minh-Thang, 148

Maamouri, Mohamed, 725
Manandhar, Suresh, 745
Manning, Christopher, 492
Maron, Yariv, 799
Martins, Andre, 34
Matsoukas, Spyros, 616
Matsubara, Shigeki, 892
Mausam, 1266
Max, Aurélien, 656
Mazur, Pawel, 913
McCallum, Andrew, 1013
McClanahan, Peter, 810
McIntosh, Tara, 356
Mehay, Dennis, 736
Mei, Qiaozhu, 1077
Mejer, Avihai, 971
Meng, Yao, 555
Mi, Haitao, 273
Mihalcea, Rada, 1057
Moldovan, Dan, 315
Moon, Taesun, 196
Mooney, Raymond, 1173
Mukherjee, Arjun, 207

Murata, Masaki, 892
Murray, Gabriel, 388

Nakov, Preslav, 148
Naseem, Tahira, 1234
Natarajan, Prem, 626
Navigli, Roberto, 116
Negi, Sumit, 87
Ng, Hwee Tou, 177, 923
Ng, Raymond, 388
Ng, See-Kiong, 218
Ng, Vincent, 229
Nie, Jian-Yun, 108
Nie, Zaiqing, 430

O'Connor, Brendan, 1277
Oates, Tim, 585
Och, Franz, 158
Ohno, Tomohiro, 892
Okumura, Manabu, 832

Paul, Michael, 66
Penn, Gerald, 23
Pereira, Fernando, 167
Persing, Isaac, 229
Petrov, Slav, 12, 167, 705
Piatko, Christine, 585
Pino, Juan, 545
Pitrelli, John, 335
Platt, John, 251
Prasad, Rohit, 626
Prasov, Zahar, 471

Qazvinian, Vahed, 1245
Qian, Longhua, 346
Qian, Xian, 187
Quinn, Alex, 127

Radev, Dragomir, 1245
Raghunathan, Karthik, 492
Rajkumar, Rajakrishnan, 564
Rambow, Owen, 1024, 1151
Rangarajan, Sudarshan, 492
Rappoport, Ari, 325, 684
Reichart, Roi, 325, 684
Reisinger, Joseph, 1173
Reiss, Frederick, 1002

Resnik, Philip, 45, 127, 284, 534
Riedel, Sebastian, 1013
Riloff, Ellen, 77
Ringgaard, Michael, 705
Ringger, Eric, 240, 810
Rosti, Antti-Veikko, 667
Roth, Dan, 767, 961, 1099
Roukos, Salim, 335
Rozovskaya, Alla, 961
Rush, Alexander M, 1
Rush, Alexander M., 1288

Sauper, Christina, 377
Scherrer, Yves, 1151
Schoenmackers, Stefan, 1088
Seppi, Kevin, 810
Setiawan, Hendra, 534
Shah, Rushin, 725
Shen, Libin, 616
Shi, Lei, 1057
Smith, Noah, 34, 1277
Song, Young-In, 430
Sontag, David, 1, 1288
Stallard, David, 626
Steedman, Mark, 575, 1223
Stiller, Stephan Hyeonjun, 1068
Su, Jian, 872
Subramaniam, L V, 87
Subramanya, Amarnag, 167
Sudoh, Katsuhito, 944
Sun, Maosong, 366
Surdeanu, Mihai, 492

Takamura, Hiroya, 832
Tan, Chew Lim, 872
Tian, Mingjun, 1057
Toutanova, Kristina, 251
Tsukada, Hajime, 944

Udupa, Raghavendra, 1256
Ungar, Lyle, 725
Uszkoreit, Jakob, 158

Vaithyanathan, Shivakumar, 1002
van Noord, Gertjan, 902

Walker, Daniel, 240
Wang, Hongling, 714
Way, Andy, 420
Weischedel, Ralph, 616
Weld, Daniel, 1088
White, Michael, 564
Wisniewski, Guillaume, 778, 933
Woodsend, Kristian, 513
Wu, Lide, 187

Xing, Eric, 34, 1140, 1277
Xu, Jinxi, 616, 667

Yan, Hongfei, 56
Yao, Limin, 1013
Yessenalina, Ainur, 1046
Yih, Wen-tau, 251
You, Gae-won, 430
Yu, Hao, 555
Yue, Yisong, 1046
Yvon, François, 778, 933

Zamparelli, Roberto, 1183
Zettlemoyer, Luke, 1223
Zhai, ChengXiang, 66
Zhang, Bing, 616
Zhang, Chen, 756
Zhang, Hui, 440
Zhang, Min, 440
Zhang, Qi, 187
Zhang, Ruiqiang, 1129
Zhang, Yue, 843
Zhao, Shaojun, 596
Zhao, Xin, 56
Zheng, Yabin, 366
Zheng, Zhaohui, 1129
Zhikov, Valentin, 832
Zhou, Bowen, 138
Zhou, Guodong, 346, 714, 882
Zhou, Ming, 1068
Zhou, Yaqian, 187
Zhu, Qiaoming, 714
Zhu, Xiaoyan, 1119
Zhu, Xingwei, 1119
Zhuang, Tao, 304
Zitouni, Imed, 335, 993
Zong, Chengqing, 304